# Fingerprint and descriptor generation - GenerateMD

This manual gives you a walk-through on how to use the GenerateMD command line tool for fingerprint and descriptor generation:



## Introduction

GenerateMD is an application program for the generation of various molecular descriptors including chemical fingerprints, namely ChemAxon Chemical Fingerprint and ECFP, as well as 2-dimensional pharmacophore fingerprints, reaction fingerprints and structural keys of

molecular structures. These descriptors can be used to characterize molecules and search through large compound libraries with respect to chemical, structural as well as pharmacological properties.

## Molecular descriptors

In our terminology the term "molecular descriptor" refers to all kinds of structural keys, hashed fingerprints, binary fingerprints, different types of pharmacophore fingerprints and scalar values. As a generic definition, a molecular descriptor is simply a *set of values associated with a two or three dimensional molecular structure*. The creation of such descriptors is called *descriptor generation*. This procedure always starts from a chemical structure, and through various feature recognition and transformations steps (which depend on the specific descriptor to be created) a series of values are generated. These values can be bits, integer or real numbers that - again, depending on the type of descriptor - can either be meaningful or meaningless themselves. For instance molecular weight (more precisely, mass) and $\log_D$ value are meaningful, they can be interpreted. On the other hand, one single bit in a hashed binary fingerprint is meaningless: the structure or properties of the original compound cannot be inferenced from this single value. It is relevant and also tangible to think of a molecular descriptor as an *encoded* (or *compressed*) representation of the chemical structure.

### Molecular descriptor sets

Molecular descriptors are powerful tools for the approximation of selected properties of chemical structures in an easy-to-handle form that allows efficient comparison and selection of compounds possessing required chemical, structural, pharmacological or biological features. These methods are not only faster than structure based approaches, where chemical structures or other high level structural models are compared against each other directly, but the accuracy of descriptor based approaches in estimating properties can be as good as the accuracy of structure based approaches.
One disadvantage of the use of descriptors, however, is that one descriptor can capture certain but apparently not an unlimited number of properties of the chemical structure encoded.

The concept of *molecular descriptor set* has been introduced in order to overcome this deficiency. A molecular descriptor set combines an arbitrary number and type of molecular descriptors into one single entity. Each component aim to depict a certain characteristic of the original compound and their combination is very information rich thus descriptor sets are even more powerful in screening large compound libraries for complex criteria than single descriptors are.

## Generating descriptors

Molecular descriptors are generated from molecular structures. Although different descriptors utilize different processing steps, still there are numerous steps common to these procedures. The building up of the molecular structure itself - which is an unavoidable initial step of all kinds of descriptor generation - provides an apparent example: it has to be loaded from either a molecular file or from a database, then appropriate sized memory structures have to allocated and initialized according to the data read. Following these costly steps the molecule has to be standardized. Standardization is another tedious process that may include aromatic ring perception, counter ion removal and various further transformations of the initial structure. The efficiency of descriptor generation can be improved if several descriptors are generated simultaneously, as the above outlined common steps are not repeated superfluously.

The command line program GenerateMD supports the generation of one descriptor in one go. This means, that multiple descriptors belonging to the same molecular structures can be generated consecutively by running GenerateMD several times. Although this is more time-consuming but much simpler to use. In this case the ease of use was deliberately chosen over efficiency. Note, however, that the Application Program Interface allows the generation of multiple descriptors at the same time.

## Storing descriptors in database

Descriptors belonging to a molecule, can be stored in relational databases. There is no limitation on the number of descriptors associated to a structure, but there is no predefined table structure which is automatically maintained by the program. Different descriptors are stored in different tables. Users are not required to be aware of details of all these. All they need to remember is the name of the structure table where molecules originated from and the name of the descriptor, given by the user itself at the time of descriptor generation.

The type of database table columns storing descriptors depends on the type of descriptor and the database management engine used. The table below summarizes supported RDBMS-s and implemented descriptors indicating the corresponding column type:

| RDBMS | Chemical fingerprint | Pharmacophore fingerprint | Feature vector |
|---|---|---|---|
| MySQL | INTEGER | BLOB | DOUBLE |
| Oracle | NUMBER(10,0) | RAW | FLOAT |
| MS SqlServer | int | IMAGE | float |
| | | | |

| | | | |
|---|---|---|---|
| InterBase | INTEGER | BLOB | DOUBLE PRECISION |
| PostgreSQL | int4 | BYTEA | float8 |
| DB2 | INTeger | BLOB(1M) | FLOAT |
| Access | INTEGER | LONGVARBINARY | DOUBLE |

Table 1 Column types of various molecular descriptors in different Relational Database Management Systems.

## Usage

The GenerateMD has two different ways you can use:

1. Passing parameters in the command line:

```
generatemd <command> [<input file>] [<options>]
```

2. Passing parameters in an XML configuration file:

```
generatemd <configuration file name>
```

These two modes are not strictly exclusive, they can be mixed various ways. Command line parameters can extend settings provided in the configuration file. The name of the input file can be specified in the command line even when parameters are defined in the configuration file, in this case the files defined in the command line are processed. However, this kind of usage is recommended only for expert users.

When specified, the configuration file must be the first argument after the `generatemd` command in the command line. The *<command>* flag is not accepted in this case.

Prepare the usage of the `GenerateMD` script or batch file as described in Preparing and Running Batch Files and Shell Scripts.

Alternatively, the `GenerateMD` class can be directly invoked:

- under Win32 / Java 2 (assuming that JChem is installed in c:\jchem):

```
java -cp "c:\jchem\lib\jchem.jar;%CLASSPATH%" \
    chemaxon.descriptors.GenerateMD <command>[<input file>]
[<options>]
```

- under Unix / Java 2 (assuming that JChem is installed in /usr/local/jchem):

```
    java -cp "/usr/local/jchem/lib/jchem.jar:$CLASSPATH" \
        chemaxon.descriptors.GenerateMD <command> [<input file>]
[<options>]
```

## Commands

```
Commands:
   c       creates new descriptor, default
   u       updates existing descriptor in database
   d       deletes existing descriptor from database
   a       adds new screening configurations to existing descriptors
   r       removes screening configurations
   l       list all screening configurations

Command syntax:
   c [input file] [options]
   u <structure table name> <descriptor name>
   d <structure table name> <descriptor name>
   a <structure table name> <descriptor name> [ <config. name>]
<config. file>
   r <structure table name> <descriptor name> <configuration name> ...
   l [ <structure table name> ]
```

The most typical use of GenerateMD is the generation of molecular descriptors. However, some other operations that are closely related to the generation of descriptors. These arise only in the case when molecular descriptors are stored in a database. Such descriptors can be manipulated with GenerateMD, moreover, it is highly recommended to avoid using general purpose database management tools since the integrity of structure tables and descriptors can easily be violated.

Descriptors generated earlier needs to be updated when new structures are imported into the structure table which the molecular descriptors are associated with. This can be done with the *u* (update) command.

Deleting old, unwanted descriptors should be done by the *d* commands (rather than issuing an SQL `DROP TABLE` statement).

Along with the descriptors generated, some other information are stored in the database: the name of the descriptor (for example pharma-for-H1), the type of the descriptor (for example PF), and the configuration used for the descriptor generation. The user of the program need to remember only the name (since it was given by him or her), although he or she should be aware of the configuration stored. Whenever the descriptors are in use (for instance used in virtual screening), this original configuration is applied. However, this may not always be sufficient, for instance new metrics might be needed for screening. Three more commands are provided to extend this original configuration.

The command *a* (add) allows the extension of the original configuration. See example how to add a new metric (optimized for another active family, but using the already generated descriptor). The configuration name given as the third parameter in the command line is stored in the database, and is used to refer to the particular screening configuration. When this name is omitted, the configuration name is taken from the configuration file name (by removing the file path and extension).

Command */* can be applied to list existing screening configurations, and *r* to remove any previously added configurations.

## Options

Options and parameters can either be defined in the command line or be specified in an XML configuration file. The command line mode is more suitable for smaller experiments. In contrast to this, configuring `generatemd` from XML is more user-friendly and convenient. However, it is only the creation of new descriptors that is available through XML configuration. In the case of the other command it would be unnecessarily time-consuming to create the configuration files instead of passing a few parameters in the command-line. There is an example configuration file available.

```
General options:
  -h, --help               this help message
  -x, --expert-help        advanced options for expert users
  -L, --list-descriptors   lists all available built-in descriptor
types
  -v, --verbose            verbose
  -s, --saveconf           saves database settings

Input/output options:
  -a, --table-name <table> takes structures from the given database
table
  -q, --query <sql>        where-clause for reading structures from a
table
  -o, --output <filepath>  SDF output file path and name
  -g, --generate-id [<first>]
                             generate unique structure identifiers
                             an optional value for the first ID can be
given
  -S, --sdf-output         generate SDfile output (otherwise
descriptor file)

Database options:
  -d, --driver <JDBC>      JDBC driver
  -u, --dburl <url>        URL of database
  -l, --login <login>      login name
  -p, --password <pwd>     password
```

```
Descriptor options:
  -k, --descriptor <type> <name> [ <comment> ]
                            generates descriptors of the given type and
name
                              in database
  -k, --descriptor <type>
                            generates descriptors of the given type in
file
  -c, --config <config>     path and name of the XML configuration file


Input/output options:
  -D, --decimal-format      tab separated output of the descriptor
values
  -2, --binary-format       0 1 string
  -T, --stat                print statistics on descriptors generated

SDfile options:
  -I, --id-tag <name>       name of the tag storing unique molecule
identifiers
  -t, --use-tag <name>      use the named tag instead of the default
one to
                              store descriptor data
  -P, --PMAP-tag [<name>]   use existing PMAP data
  -w, --validate <name>     validate the descriptor using activity-
seeded,
                              structure-based clustering, <name> is an
SDfile
                              tag containing the IC50 activity data

Database options:
  -O, --proptable [<table>]  name of the property table
  -U, --update-on-insert    update descriptors when new structure is
inserted

Chemical fingerprint options:
  -f, --fingerprint-length chemical fingerprint length in bits
  -n, --pattern-length      maximum number of bonds in patterns
  -b, --bit-count           maximum number of bits to set for each
patterns

Fuzzy smoothing options:
  -z, --fuzzy <smoothing factor>
                            generate fuzzy pharmacophore fingerprints
with the
                              given fuzzy smoothing factor
  -G, --Gaussian-cutoff <value>
                            not smoothing outside <value>*sigma
  -B, --smoothing-bound <value>
                            lower bound for fuzzy smoothing
  -A, --asymmetrical-smoothing
```

```
                              do not smooth left side
  -R, --ignore-rotomers       do not take into account the effect of
rotatable
                                bonds
```

Merging short forms of command line options is not supported (that is, instead of *-Tv* the form *-T -v* should be used).

In the XML configuration file the same parameters as in the command line can be defined. The command line options appear in the *config* configuration editor labeled with the above long forms of command line parameter names, with only small differences. In most of the cases only the first letters of words are capitalized. For example *--asymmetrical-smoothing* is displayed as *AsymmetricalSmoothing*.

> To use GenerateMD a valid license key is needed. When no valid license key is found in the home directory, GenerateMD runs in demo mode, where the number of molecular descriptors to be processed is limited to 2000 (thus if several types of molecular descriptors are generated, then the number of structures may be limited to few hundreds).

## Input

The software may take molecules from either a text file or from a database. The input file can be specified following the command type. If the file name is omitted, the standard input is read, except when the option -q, *--query* is defined in which case input is taken from a database. Input files should be molecular structure files of one of the common formats like MDL molfile, Compressed molfile, SDfile, Compressed SDfile, SMILES (see the Administration Guide of JChem for a full list of molecular structure files accepted).

If the input file is an SDfile, it may already contain the descriptor (or some of them) to be generated, or other data needed for descriptor generation (for instance, the pharmacophore map (the list of pharmacophore points) which is required for pharmacophore fingerprint generation). Such existing data can either be used or omitted during descriptor generation. The default behavior of GenerateMD is to omit such information. In order to use it the -P, --PMAP-tag flag has to be specified. The default SDfile tag for the pharmacophore map is PMAP. Other than default tag name can be specified after the *-P, --PMAP-tag* option.

## Output

GenerateMD is capable of writing result descriptor sets either into a text file (*-o, --output*) or into a database table when database options are defined (either explicitly by the appropriate flags or

implicitly by a JChem user configuration file `.jchem` in the user's home directory). If no output is specified, results are written to standard output as readable text.

## Descriptors

When creating or updating descriptors, one *-k* flag is mandatory. Descriptor type specific flags always follow the this flag. GenerateMD supports the following descriptor types: CF, PF, Burden eigenvalue descriptor (or BCUT) and various scalar descriptors. Besides, the generation of user-defined custom descriptors is also supported.

### Chemical fingerprint

Chemical fingerprints encode the topological structure of molecular structures into a bit string. This fingerprint is a hashed binary fingerprint of a given length. The length (number of bits in the fingerprint) can be set either in an XML configuration file or in the command line with the *-f (--fingerprint-length*) flag. If both the configuration file and the command-line specifies the length of the fingerprint, the one given in the command-line supresses the other and takes effect. The length parameter must be a multiple of 32. Other two parameters are related to the fingerprint generation, the number of bonds in patterns and the number of bits to be set. Both can be specified wither in XML or in the command-line.  Please note that in the case of chemical fingerprints the XML configuraton file is optional.

### Reaction fingerprint

Reaction fingerprint RF are used in reaction searching and reaction similarity searching. This fingerprint is a segemented fingerprint that is constituted from 8 chemical fingerprint sections. The length of the fingerprint is 2048 by default, though this can be modified in the XML configuration file. However, the size of the segments cannot be changed individually, those are always proportional to the total length of the fingerprint as described here.

### PF: 2-dimensional pharmacophore fingerprint

Pharmacophore fingerprint generation requires the definitions of pharmacophore point types that are given in a configuration file along with many other parameter settings. The name and path of this file is given after the mandatory flag *-c, --config*.

A related option is *-P, --PMAP-tag*, which specifies the name of the pharmacophore map tag in the SDfile input. When this flag is given, the existing pharmacophore map data is taken from the input file (instead of generating it from the molecule according to the settings in pharmacophore fingerprint configuration file). Note, that the consistency of the data stored in PMAP tag, and the configuration specified after the *-c* flag has to be ensured, otherwise proper operation cannot be guaranteed.

Both pharmacophore map and pharmacophore fingerprint data are also saved in the output file, default tag names are PMAP and PF, respectively. These names can be changed by specifying the *--PMAP-tag* and *--use-tag PF* options. If the only output required is the molecular descriptors generated, then the *--descriptors-only* flag has to be specified.

Pharmacophore map data (value associated with the PMAP tag) can be visualized using MarvinView. In order to do so a palette definition property file has to be provided using the *-p* flag for mview and the name of the pharmacophore map flag (which is PMAP by default) has to be specified using the *-t* flag.

**Fuzzy pharmacophore fingerprint**

The modeling capabilities of pharmacophore fingerprints can be enhanced with *fuzzy smoothing*. The idea of that is based on the following consideration: imagine a structure where two acceptor atoms are three bonds apart on a chain, and another compound in which to acceptors are 4 bonds apart, again, on a chain. As these two chains may exhibit substantial rotational freedom, the two compounds can easily bind to the same acceptor sites in their receptor. Yet, the two fingerprints (e.g. 0,0,1,0,0,0 and 0,0,0,1,0,0) show a great degree of dissimilarity. The poor representation of the geometrical distance with the use of topological distance (number of bonds along the shortest path) in pharmacophore fingerprints can be improved by smoothing histogram bars. This means, that instead of incrementing a bin by 1, when a feature pair is found a given distance away from each other, a smaller value, let say 0.8 is put in the corresponding bin, while the remaining 0.2 is divided between the to neighboring bins equally. This way one would get the following two fingerprints: 0,0.1,0.8,0.1,0,0 and 0,0,0.1,0.8,0.1,0 which give a better representation of a potential geometrical arrangements of the two acceptor points.

Fuzzy smoothing fits a normal distribution to each histogram bins and this way a 'sharp' histogram bars are smoothed, made slightly flatter and in the case of near misses slightly more similar. The degree of smoothing (the standard deviation of the normal distribution, specified following the *-z* flag) depends on the number of rotatable bonds between the two pharmacophoric points: the larger the number of rotatable bonds the larger the standard deviation of the distribution, thus one feature pair at a larger distance away from each other contribute to more neighboring bins. This affect of rotatable bonds can be ignored by specifying the *-R* flag. A more explicit control of the distance dependency can be achieved by the *-B* flag which enable the specification of the smallest topological distance from which fuzzy smoothing should be applied. Point pairs closer to each other than this limit will not be smoothed.

By default, smoothing spans over the entire histogram, however, farther away from the central cell (which the pharmacophore point pair in question would belong to) the effect of smoothing is questionable: values of the normal distribution are very small which are better to be ignored. This can be controlled by the *-G*, with this a multiple of  can be defined beyond which no spreading of fuzzy data is done.

A further option is to apply smoothing only toward cells representing larger distance than the central cell (that is, only the right side of the Gaussian curve is considered).

Fuzzy smoothing is a useful tool to generate hypotheses too, since in the 'sharp' case it is often experienced that important columns completely disappear from the common histogram due to

the above described 'near miss' phenomenon. The fuzzy hypothesis, though often quite flat still contains small values at some specific distances, thus they in some cases give better hits in screening.

### Burden eigenvalue descriptors

The three standard BCUT descriptor types are supported: charge, polarizability and hydrogen bonding properties. Besides the classic Burden-matrix is also available. Both lowest and highest eigenvalues are calculated. By default the two lowest and the two highest eigenvalues of the Burden-matrix (connectivity matrix weighted by atom number) are calculated. Default settings can be overriden in a configuration file .

### Scalar descriptors

Scalar descriptors are single values associated with an entire molecular structure. Molecular mass, number of atoms provide the simplest examples. Others are more complex in nature and may involve the estimation of physico-chemical properties. For instance log*P*, log*D* and topological polar surface area (TPSA) are members of this class of descriptors.

Some of the scalar descriptors take a few optional parameters, while others do not need any external parametrization. In this latter case the *-c* flag is always omitted. *-c* is always optional, if omitted for descriptors that require parameters, default values are used.

To obtain the list of available descriptors call GenerateMD with the *-L, --list-descriptors* flag.

## File output

In the case of SDfile input an SDF output can be generated by the use of the *-S* flag. Otherwise, a *descriptor file* is created. Such descriptor files are readable text files that can be processed by other applications, e.g. by ScreenMD. Descriptor files contain one descriptor per line along with a unique identifier. The first two lines form the file header that should be ignored when the file is read by users.

Descriptors generated by GenerateMD can be used in JKlustor, too. To produce descriptors in a format accepted by JKlustor tools, the decimal output format options has to be specified with the *-D* flag. Using *-D* does not imply *-g*, i.e. unique identifiers are not generated, and *-g* has to be set explicitely if required. Besides the automatically generated identifiers, natural identifiers given in the input SDfile can also be printed in the JKlustor typed output file. For this purpose the flag *-I, --id-tag* can be used.

## Database output

Descriptors generated can be stored in database tables. The appropriate table is created and maintained automatically. The user of the program should only be aware of the name of the descriptor given at the time of generating the descriptor (second parameter after the *-k* flag).

This name is used to access the descriptor data from other applications. Descriptors can either be inserted into empty rows (command *c*) or alternatively existing data can be updated (command *u*).

An important database related option is *-U*, *--update-on-insert*. Descriptors created with this option selected will be updated (i.e. generated) whenever a new structure is inserted into the corresponding structure table. If this option is not specified in the command line, the descriptor table is not updated automatically, however, at each use (e.g. similarity searching) a warning message is given that the regeneration of the descriptors is required because the structure table contains newer structures.

## Statistics

Simple statistics on the generated descriptors can also be obtained as an output of GenerateMD by specifying the *-T* (*--stat*) option. This is always written to the standard output. It serves as a first-hand aid to evaluate and tune descriptor parameters.

## Configuration Files

Besides the XML configuration file that can be optionally used to specify parameter settings, the GenerateMD application may take other configuration files too. Such files correspond to the molecular descriptors to be generated.

Different descriptor types require different parametrization. The actual parameter settings are defined in external text (XML) files. One such file corresponds to one descriptor type.

### Chemical fingerprints

Since there are only three parameters that determine the chemical (topological) fingerprint, the corresponding configuration file is very simple. Two sections are related to the generation of fingerprints, `<StandardizerConfiguration>` and `<Parameters>`. This latter can have three attributes, `Length`, `BondCount`, `BitCount`.

#### Example

```
<Parameters Length="1024" BondCount="7" BitCount="3"/>
```

The values above are the default settings.

## 2-d pharmacophore fingerprints

The pharmacophore fingerprint configuration file is shared among various components of pharmacophore tool-set. Detailed description of the relevant sections of the file is given in the most relevant document: the first two sections correspond to PMapper, the third introduces pharmacophore fingerprint parameters, thus it is described in this document, while the fourth section defines parameters that are used by ScreenMD.

The third section of the pharmacophore configuration file is `<PharmacophoreFingerprintParameters>`. Parameters defined in this section influence pharmacophore fingerprint generation, but not the pharmacophore point perception or dissimilarity calculations.

**Example**

```
<PharmacophoreFingerprintParameters
       MinimalDistance="1"
       MaximalDistance="10"
       FuzzySmoothingFactor="0.7"
       />
```

The first two parameters specify the minimal and maximal topological distance between pharmacophoric points. These two parameters are mandatory, since that fundamentally determine the pharmacophore fingerprints generated. The third parameter is optional, when specified, fuzzy pharmacophore fingerprints are generated with the given smoothing factor.

## BCUT descriptors

The configuration file determines which variety of the Burden eigenvalue descriptors is calculated. Four types are available:

i. Burden, the weighted connectivity matrix with atomic numbers in the diagonal.
ii. Charge, the Burden-matrix with partial atomic charges in the diagonal.
iii. Polarizability, the Burden-matrix with estimated polarizability in the diagonal.
iv. HBond, the Burden-matrix with estimated hydrogen bonding properties, the diagonal contains the number of acceptor and donor sites found in the macro-state of the molecule.

The parameter file specifies which eigenvalues of the modified connectivity matrix is computed. Either lowest or highest, or both lowest and highest eigenvalues, one or more of them.

**Example**

```
<BCUTDescriptors>
        <BCUT Type="Burden" Lowest="2" Highest="2" />
    </BCUTDescriptors>
```

One configuration file can define multiple descriptors, for instance the standard BCUT (as described in the publications) can be defined as follows:

```
<BCUTDescriptors>
        <BCUT Type="Charge" Lowest="1" Highest="1" />
        <BCUT Type="Polarizability" Lowest="1" Highest="1" />
        <BCUT Type="HBond" Lowest="1" Highest="1" />
    </BCUTDescriptors>
```

## Examples

The following examples are more complex ones, with detailed explanation:

- Example of usage with a configuration file in Windows:

  ```
  generatemd config examples\config\generatemd.xml
  ```

  The example configuration file sets all required parameters, specifies input and output files.

- A UNIX command that reads molecules from file `nci1000.smiles` and writes results in the file named `nci1000.pf`:

  ```
  cd examples
  generatemd c molecules/nci1000.smiles -k PF -c config
  /pharma-frag.xml -o nci1000.pf
  ```

  One fingerprint per line is written to output (which can result in very long lines). The first three fingerprints in the output:

  ```
  1 a a = | 0 0 0 0 1 0 0 0 0 0 |, h a = | 2 4 5 3 0 0 0 0 0
  0 |, h h = | 7 8 5 1 0 0 0 0 0 0 | 2 a a = | 1 6 4 0 4 0 0
  0 0 0 |, h a = | 10 10 12 16 8 12 8 8 0 0 |, h h = | 12 16
  11 4 4 4 8 8 12 12 | 3 a a = | 0 2 0 2 0 6 0 0 0 0 |, d a
  = | 0 0 0 2 0 2 0 0 0 0 |, h a = | 1 6 11 9 8 0 0 0 0 0 |,
  ```

```
h d = | 1 2 3 1 0 0 0 0 0 0 |, h h = | 7 8 5 1 0 0 0 0 0 0
|
```

- Processing an SDfile, generating pharmacophore fingerprints and storing them in an other SDfile along with the original molecules and tags given in the input file:

```
cd examples
generatemd c molecules/nci1000.sdf -o nci1000.sdf -S -
k PF -c config/pharma-frag.xml
```

The output file is also in SDfile format, but it contains two new tags, PMAP and PF. In the case of the first structure these are as follows:

```
> <PMAP> h;h;h;a;h;h;h;a > <PF> a a = | 0 0 0 0 1 0 0 0
0 0 |, h a = | 2 4 5 3 0 0 0 0 0 0 |, h h = | 7 8 5 1 0 0
0 0 0 0 |
```

- Similar to the one above, but this time only descriptors are written, and unique identifiers are generated and stored in the output file:

```
cd examples
generatemd c molecules/nci1000.sdf -k PF -c config
/pharma-frag.xml -g -o nci1000.pf
```

The output file is a descriptor-set file.

- Generating reaction fingerprints for each reaction in the given input file:

```
generatemd c molecules/reactions.rdf -k RF -c config
/rfp.xml -o reactions+fingerprint.sdf -S
```

The output is an SDFile, that contains the original reaction along with the reaction fingerprints (in readable string format).

- Pharmacophore fingerprints generation, but this time structures are taken from a database, thus descriptors are also stored in a database table. The table for molecular structures is *my_mols*:

```
generatemd c -a my_mols \
    -k PF pharma_for_H1 "these descriptors perform
very well in screening for H1 agonists" -c pharma-frag.
xml  \
    -d "org.gjt.mm.mysql.Driver" -u "jdbc:
mysql://localhost/cdexample" -l cduser -p secret
```

Note, that the *-a, -d, -u, -l, -p* flags are used always together, and if one of them is given in a command line, all others must be specified, too. Read more about how to set the JDBC driver and the URL of the database. The parameters can be stored in the `.jchem` file with the use of *-s, --saveconf* flag. When stored, these values are retrieved on start-up of the program and the *-d , -u, -l, -p* flags are not needed.
When calling GenerateMD, the database engine should run, the table to be inserted fingerprints into should exists, the JDBC driver has to be installed on the local host, and the user *cduser* has to be an existing one.

Also note, that following the descriptor type after the *-k* flag a symbolic name, given by the user (*pharma_for_H1* in this example, referring to the pharmacophore point type defined in the configuration used (acceptor, donor, hydrophobic, aromatic, plus, minus)) and an arbitrary remark can also be specified. Both of these are optional, however, greatly help the user when navigating through a large set of different kind of descriptors associated with molecular structures.

- Adding two new parametrized metrics (or generally new screening configurations) to an existing descriptor *pharma_for_H1*:

```
generatemd a my_mols pharma_for_H1 "Asymmetric
Euclidean with very high Enrichment" veryselective.xml
generatemd a my_mols pharma_for_H1 "Weighted Euclidean
for scaffold hopping" scaffoldhopp.xml
```

These two screening configurations (*veryselective.xml* and *scaffoldhopp.xml)* contain one or more metrics each, and should be created manually. All of these screening configurations then can be used in virtual screening applications.

- Listing all descriptors stored in the database:

```
generatemd l
```

The output of this command is:

```
Structure table: my_mols
Descriptor name: pharma_for_H1
Descriptor type: PF
Comments: these descriptors perform very well in screening
for H1 agonists
```

- Removing the descriptors generated in the example above:

```
generatemd d my_mols pharma_for_H1
```

When executed, this command deletes the entire table that stored *adhrpm* and all other administrative data.

- The next example demonstrates the use of descriptors with JKlustor tools:

```
generatemd c molecules/nci1000.sdf -k PF -c config
/pharma-frag.xml -D -g -v -o ward-input.txt
ward -f 0 -m 210 -c 10 -i ward-input.txt -o ward-
result.txt
```

With these commands 1000 structures from the input file `molecules /nci1000.sdf` are clustered (using Ward clustering) into 10 clusters based on pharmacophore similarity/dissimilarity.

SDfiles containing the pharmacophore map tag can be generated by the PMapper tool. Although the pharmacophore fingerprint generation procedure requires not only the pharmacophore map, but the original molecular structure, too, it is still worth preparing SDfiles with the *PMAP* in case when different parameter settings (bin size, minimal-, maximal- distance) for the fingerprint generation are tested, since generating pharmacophore map takes significantly longer, than the rest of the fingerprint generation process.

## References

1. Schneider, G.; Clement-Chomienne, O.; Hilfiger, L.; Schneider, P.;Kirsch, S.; Bohm, H-J. and Neihart, W. Virtual Screening for Bioactive Molecules by Evolutionary De Novo Design *Angew. Chem. Int. Ed.* 2000, 39, 4130-4133
2. Schneider, G.; Lee, M-L.; Stal, M. and Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks *J. Comp-Aid. Mol. Des.* 2000, 14, 487-494
3. Pearlman, S. R. and Smith, K. M. Novel Software Tools for Chemical Diversity, Perspectives in Drug Discovery and Design, 9/10/11: 339-353, 1998.
4. Burden, F. R. Molecular identification number for substructure searches, J. Chem. Inf. Comput. Sci. 29 (1989), 225-7.

## Legal notes