

# Technical Report

Department of Computer Science  
and Engineering  
University of Minnesota  
4-192 EECS Building  
200 Union Street SE  
Minneapolis, MN 55455-0159 USA

TR 08-020

AFGEN2.0

George Karypis

June 24, 2008



AFGEN is a program that generates descriptor spaces for chemical compound(s). The descriptor space consists of graph fragments that can have three different types of topologies: paths (PF), acyclic subgraphs (AF), and arbitrary topology subgraphs (GF).

This manual is divided in the following sections:

- [Installation](#)
- [Running AFGEN](#)
- [Input Files](#)
- [Output Files](#)
- [Examples](#)
- [Limits](#)
- [Credits & Contact Information](#)
- [Copyright Information](#)

---

## Installation

AFGEN is provided as a binary distribution with pre-built executables for Linux (both for 32 and 64 bit architectures). Additional binaries can be provided upon request.

Once you download AFGEN, you need to uncompress and untar it using the following commands:

```
> tar -xzf afgen-2.0.tar.gz
```

This will create a directory named `afgen-2.0` with the following structure:

```
afgen-2.0\  
  builds\  
    Linux-i686\  
    Linux-x86_64\  
  doc\  
  examples\  

```

---

## Running AFGEN

The name of the AFGEN executable is `afgen` and is located in the architecture-specific subdirectory under `builds`. The `afgen` program is invoked at the command-line within a shell window (e.g., `xterm`, `Gnome terminal`, etc).

### Operational Modes

The `afgen` program can be used in two different modes. In the first mode, referred to as the *descriptor-space generation mode (DSGM)*, `afgen` reads a library of compounds,

finds all the fragment-based descriptors, and then represents each library compound as a frequency vector in that descriptor space.

In the second mode, referred to as the *descriptor-space projection mode (DSPM)*, `afgen` takes as input a library of compounds **and** a previously generated set of `afgen` descriptors (using the `-fragfile` option) and then represents each library compound as a frequency vector in that descriptor space. Note that in this second mode, no new fragments (i.e., descriptors) are generated.

## Manpage

The manpage for `afgen` is the following (can be obtained by typing `afgen -help`):

```
Usage
  afgen [options] filename

Required parameters
  filename
    The file containing the library of compounds. The
    compounds are
      specified using MDL's SD file format
      (http://www.mdl.com) or Tripos's
      mol2 format (http://www.tripos.com).
    The specific format of the library is determined by the
    extension of
      the file, which can be either .sdf or .mol2.

Optional parameters
  -fragtype=string
    Specifies the type of fragment-based descriptors to be
    generated.
    Supported options are:
      gf  arbitrary fragments [default]
      af  acyclic fragments
      pf  path fragments

  -lmin=int
    Specifies the minimum length (in terms of bonds) of
    the generated
      fragments. The default value is 3.

  -lmax=int
    Specifies the maximum length (in terms of bonds) of
    the generated
      fragments. The default value is 7.
```

`-fmin=int`

Specifies the minimum frequency that a fragment must have before

it becomes a descriptor. The frequency of a fragment is based on

the number of distinct compounds that it occurs at. The default

value is 1 (i.e., all fragments are treated as descriptors).

`-noh`

This option forces `afgen` to remove any hydrogen atoms from the compounds.

`-noatyping`

Forces `afgen` to ignore the atom typing specified in the input file

(if any). If this option is used, then only the basic atom types are

used (e.g., P, N, O, etc.). This option applies only to inputs files

that use the mol2 format.

`-armark`

Detects aromatic rings in SDF files and relabels the bonds as aromatic.

`-outfile=string`

Specifies the stem of the output files (.out & .frags.[sdf, mol2]).

If outfile is not specified then the output stem is the same as input

stem.

`-fragfile=string`

Specifies the file containing previously generated fragments to be used

for restricting the descriptor representation of the library. If this

file is specified, the compounds will be represented as vectors in

that descriptor space and any fragments contained in the compounds

but not in that file will be ignored.

`-nooutput`

Does not produce any output files and used for testing only.

```
-help
```

```
Prints this message.
```

---

## Input Files

The `afgen` program uses two different input files. The first is the file that stores the compound library. At this point `afgen` supports library files in MDL's SD file format (`.sdf` extension) and Tripos's Mol2 file format (`.mol2` extension). These are two of the most widely used structural formats for chemical compounds. Information regarding the SDF format can be found at <http://www.mdol.com> and information regarding the Mol2 format can be found at <http://www.tripos.com>. If your library is not already in SDF or Mol2 formats, you can use OpenBabel <http://openbabel.org> to convert your input files in one of these two formats.

The second file, supplied via the `-fragfile` optional parameter, is the fragment file generated from a previous run of `afgen`. You should not manually edit this file, unless you have a good understanding of the information stored in it.

---

## Output Files

Depending of the operational model of `afgen`, it will create either two or one files. When `afgen` is used to generate a set of fragment-based descriptors for a library (**DSGM**) it will produce both a fragment file (extension `.frag.sdf` or `.frag.mol2`) containing the SDF/Mol2 representation of the discovered fragments (i.e., descriptors) and the descriptor-based representation of the library (extension `.out`). When `afgen` is used to generate the descriptor-based representation of a library with respect to a previously discovered set of graph fragments (**DSPM**), it will only produce the descriptor-based representation of the library.

### Note:

In addition to these files, while running, `afgen` creates a temporary file with the extension `.afgen-tmp`. Upon successful completion, `afgen` deletes this file.

## Format of the Fragment File

The following shows the first few lines of the fragment file generated by executing:

```
afgen -lmin=2 -lmax=3 test1.sdf
```

(The `test1.sdf` file is in the `examples` directory).

```
AFGEN
20AFGEN-GF:2-3:1
```

```
1 0 0 0 0 0 0 0 0999 V2000
0.0 0.0 0.0 C 0 0 0 0 0
M END
> <AFGEN_NFRAGS>
19
> <AFGEN_VLBLs>
3
000H
001C
002O
$$$$
1
20AFGEN-GF:2-3:1
3 2 0 0 0 0 0 0999 V2000
0.0 0.0 0.0 C 0 0 0 0 0
0.0 0.0 0.0 C 0 0 0 0 0
0.0 0.0 0.0 C 0 0 0 0 0
1 2 1 0 0 0 0
1 3 2 0 0 0 0
M END
> <AFGEN_EDGES>
2
0:1:1:1:0
0:1:2:1:1
$$$$
2
20AFGEN-GF:2-3:1
4 3 0 0 0 0 0 0999 V2000
0.0 0.0 0.0 C 0 0 0 0 0
0.0 0.0 0.0 C 0 0 0 0 0
0.0 0.0 0.0 C 0 0 0 0 0
0.0 0.0 0.0 C 0 0 0 0 0
1 2 2 0 0 0 0
1 3 1 0 0 0 0
2 4 1 0 0 0 0
M END
> <AFGEN_EDGES>
3
0:1:1:1:1
```

```
0:1:2:1:0
```

```
1:1:3:1:0
```

```
$$$$
```

(view [test1.sdf](#) and [test1.frag.sdf](#))

The format of the fragment file follows the format of the input file. If the input library is in SDF format then the fragment file is in SDF format. If the input library is in mol2 format, then the fragment file is in mol2 format.

The above example shows the SDF-version of the fragment file. The first compound corresponds to a dummy compound that contains some information related to the number of discovered fragments (`AFGEN_NFRAGS`) and a mapping of the chemical atoms to internal numbering (`AFGEN_VLBLs`). The rest of the file contains the SDF format of the fragments that were generated numbered from 1. This fragment numbering is used by the descriptor-based representation file to indicate the set of fragments along with their frequencies that are contained in each compound.

The mol2-version of the fragment file contains similar information.

**Note:**

Besides the standard atoms/bonds information for each fragment, the file also contains the list of bonds in a canonicalized form (`AFGEN_EDGES`). This information are used by `afgen` when operating in the **DSPM** mode and should not be modified.

### Format of the Descriptor-Space Representation File

The following shows the first few lines of the descriptor-space compound representation file generated by executing: `afgen -lmin=3 -lmax=4 -fmin=30 test2.sdf`.

```
>318
1:5 2:2 3:2 4:9 5:4 6:1 7:1 8:2 9:2 10:1 11:1 12:2 13:2
14:2 15:2 16:1 17:2 18:2 19:1
3:3 4:11
>332
1:4 2:2 3:2 4:7 5:3 6:1 7:1 8:2 9:2 10:1 11:1 12:1 13:1
14:1 15:1 16:1 20:1 21:1
3:1 4:10
>432
1:4 4:7 5:3 10:1 11:1 13:1 14:2 15:2 16:1 20:2 22:1 23:1
24:1 25:2 26:1 27:2 28:1 29:5 30:6 31:1
3:10 4:20
>656
1:6 4:8 5:3 10:4 11:2 13:2 14:2 15:2 29:1 32:2 33:2 34:2
35:2 36:2 37:2 38:2 39:2 40:1 41:1 42:1 43:1
3:5 4:12
>836
```



```
1:5 2:1 3:1 4:5 5:2 10:1 11:1 13:1 14:6 15:6 17:1 18:2 19:2
20:1 21:3 25:5 26:1 27:3 31:3 44:2 45:1
```

```
3:3 4:9
```

```
>847
```

```
18:3 20:4 21:13 22:4 24:4 25:5 28:5 29:8 30:3 31:1 33:2
35:1 37:1 40:2 44:5 45:3 46:2 47:1 48:2 49:2 50:2 51:2
3:77 4:167
```

```
>849
```

```
18:15 21:27 25:9 31:6 44:21
3:38 4:125
```

```
>851
```

```
1:4 2:1 3:1 4:7 5:3 6:1 7:1 8:2 9:2 10:1 11:1 12:1 13:1
14:2 15:2 20:1 21:1 25:2 26:1 27:2 31:1 44:1 45:1
3:1 4:3
```

(view [test2.sdf](#), [test2.frag.sdf](#), and [test2.out](#))

The descriptor-space representation file contains three lines for each library compound. The first line (starting with a > character) contains the name of the compound and is obtained from the library file itself.

The second line contains the actual descriptor-space representation. It consists of a sequence of `fragment-ID:frequency` pairs separated by a space (sorted in increasing fragment-ID order). The fragment-ID corresponds to the name of the fragment as stored in the fragment file and the frequency counts the number of embeddings of this fragment in the compound.

The third line contains a sequence of `fragment-length:frequency` pairs for those descriptors that are not included in the descriptor-based representation of the compound. These numbers will be non-zero under two scenarios. First, if a value greater than 1 is specified for `-fmin`, some of the fragments present in the library may be pruned because they do not meet the minimum frequency cutoff. For each fragment that is pruned, their frequencies are added up on a per-length basis and are reported in this line. For example, in the case of the 318 compound in the above example, due to the `-fmin=30`, it resulted in pruning fragments of length 3 whose total frequency was 3, and fragments of length 4 whose total frequency was 11. The second case in which these frequencies will be non zero are when `afgen` is used in the **DSPM** mode. Since the descriptors are restricted to only those provided by the `-fragfile` option, some of the fragments present in each compound may not have corresponding descriptors. In such cases, the total frequency (on a per fragment length basis) for these ignored fragments is reported in that line.

---

## Examples

The following shows the information `afgen` outputs to the screen by executing: `afgen -lmin=3 -lmax=4 -fmin=30 test2.sdf`.

```

*****
*****
AFGEN 2.0.0 Copyright 2006-2008, Regents of the
University of Minnesota

(HEAD: 3939, Built on: Mar 21 2008, 11:22:29)

Library Information -----
-----
Library file: test2.sdf
#Compounds:    99, #AtomTypes:  10, #EdgeTypes:   8

Options -----
-----
fragtype=GF, lmin=3, lmax=4, fmin=30, noh: no
outfile: test2, fragfile: <not specified>

Generating fragments... -----
-----
nfrags: 572, nslfrags: 884, lnnz: 5594, ncfra
gs: 0, hr: 92.94%
Done.

Pruning fragments... -----
-----
pnfrags: 53
Done.

Writing fragments ... -----
-----
Done.
*****
*****

```

Among the information that is printed the most important are that number of fragments that were discovered (`nfrags`) and the number of non-zeros in the descriptor-based representation of the library (`lnnz`). The number of non-zeros indicates the space that will be required to store the descriptor-based representation in memory.

Among the other displayed information, `ncfrags` is the number of fragments read from the file specified via `-fragfile` and `pnfrags` is the number of fragments after frequency-based pruning.

---

## Limits

The following is a set of limits on the structure/size of the compounds and fragments imposed in the current version of `afgen`:

Property	Limit
Number of atoms in a compound	511
Number of bonds in a compound	511
Number of atoms in a fragment	32
Number of bonds in a fragment	32
Degree of an atom in a fragment	5
Number of atoms types	256
Number of bond types	8

---

## Credits & Contact Information

AFGEN was written by Nikil Wale (version 1.0) and George Karypis (version 2.0).

If you encounter any problems or have any suggestions, please contact George Karypis via email at [karypis@cs.umn.edu](mailto:karypis@cs.umn.edu).

---

## Copyright Information

Copyright and License Notice

-----  
The AFGEN package is copyrighted by the Regents of the University of Minnesota.

It can be freely used for educational and research purposes by non-profit institutions and US government agencies only. Other organizations are allowed to use AFGEN only for evaluation purposes, and any further uses will require prior approval. The software may not be sold or redistributed without prior approval. One may make copies of the software for their use provided that the copies, are not sold or distributed, are used under the same terms and conditions.

As unestablished research software, this code is provided on an ``as is'' basis without warranty of any kind, either expressed or implied. The downloading, or executing any part of this software constitutes an implicit agreement to these

terms. These terms and conditions are subject to change at any time without prior notice.