FastGrow Command Line Documentation Version 1.4

Sascha Jung, Marcus Gastreich & Florian Flachsenberg

December 18, 2024

Contents

1 Introduction **2**

2 Technical Prerequisites **4**
2.1 Required Software . . . 4
2.2 Licensing . . . 4

3 Jump Start: Structure-based Fragment Growing **6**

4 Command Line Options **7**
4.1 Overview . . . 7
4.2 Program Options . . . 8

5 Generating Input Files for FastGrow **10**
5.1 Preparing a FastGrow definition file . . . 10
5.2 Preparing a core ligand file . . . 10
5.3 Preparing a FastGrow database file . . . 11
5.3.1 FastGrowDBCreator: Help . . . 13
5.3.2 FastGrowDBCreator: Program Options . . . 13

6 General Options **15**

7 Further Reading, References **16**

1 Introduction

All links, references, table of contents lines etc. in this document are clickable.

FastGrow is a powerful command line tool for structure-based fragment growing. The shape-based descriptors and efficient algorithms behind it allow very fast, explorative screenings of fragment and building block libraries and the efficient growing from a core molecule into the binding site of a protein. FastGrow is also a "component" within our flagship 3D modeling platform SeeSAR that has been conceived for drug researchers of any discipline and educational level. Within SeeSAR's Inspirator Mode, you can conveniently define the attachment point and direction for growing (the exit vector). Additionally, you can easily define pharmacophore constraints, interactively inspect and post-process the results, e.g. by scoring the growing solutions with our HYDE scoring function. For the first time, the lightning speed of FastGrow lets you grow a fragment or ligand in the binding site of its protein on-the-fly, screening thousands of fragments and building blocks within seconds.

FastGrow on the command line features:

- Structure-based growing from a fragment or ligand bound to a binding pocket
- The screening of large, pre-computed databases containing fragments or building blocks within seconds
- Output up to 100,000 growing solutions in a single run
- Straightforward integration into automated drug design workflows
- Control over clashes, pose optimization, and removal of reactive groups

Please note that this package is a command line package.

Background FastGrow is an efficient tool for the structure-based directional growing of a fragment or core ligand (both called core in the following) bound in the pocket of a protein. The main use case is the attachment of suitable fragments or building blocks (contained in a dedicated database, so called FastGrow database) to a core present in a binding pocket of a protein. Thereby, the attachment point and growing direction can be defined by choosing the appropriate bond of the core to which the new fragment should be attached (we call these "exit vectors"). The FastGrow method is based upon cylindrical shape descriptors (ray volume matrices, RVM descriptor) describing the free and occupied space of the pocket in the growing direction (see Figure 1A).[1] The corresponding shape descriptors for the fragments or building blocks are stored in the pre-processed FastGrow database (see Figure 1B). The shape descriptors are optimized for a very fast shape matching. The enormous speed comes from bit-shifts that simulate the rotation. For the FastGrow screening, the pocket descriptor is used as a query and matched against the pre-calculated shape descriptors of the fragments and building blocks in the database. These are scored and ranked by their steric fit. The top-ranked fragments from the database are attached to the core. The poses are further refined and filtered for remaining clashes and unwanted reactive groups that may have formed upon the attachment of the fragment to the core. Those compounds which survive the post filtering process are reported to the user.
For a more detailed description of the shape-based algorithm behind FastGrow, see the original publication by Penner et al.[1] The shape-based descriptors perform very well in several self-growing and cross-crowing scenarios, reproducing the crystal structure conformation in up to 84% of the cases with less than 2 Å deviation.

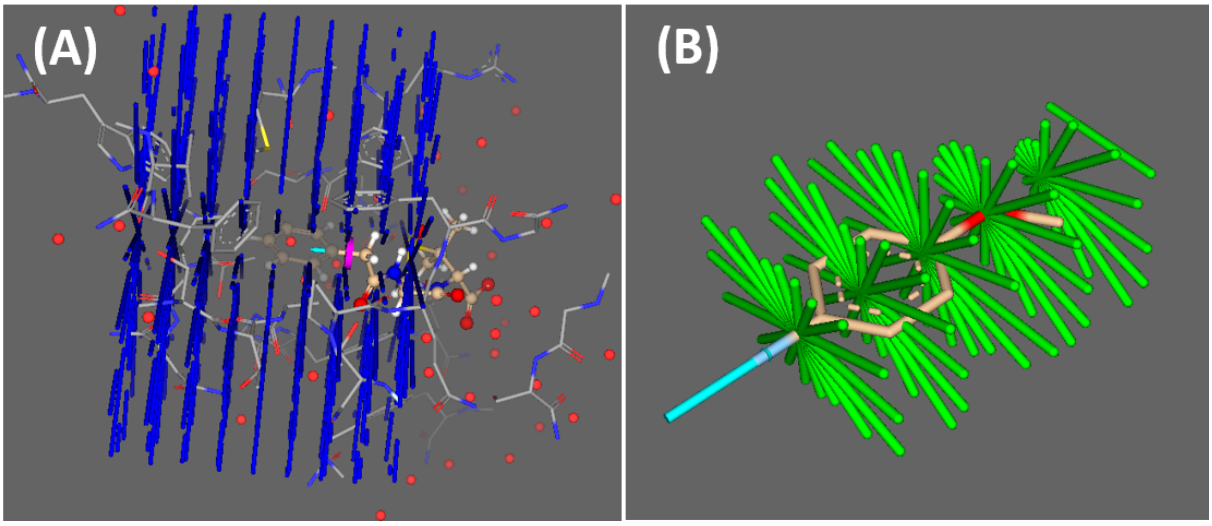

Figure 1: *(A) Cylindric descriptor (sampling rays) for the binding pocket of a protein in direction of the exit vector (cyan arrow) defined alongside a bond of the bound fragment. (B) Example for a descriptor of a single conformer of a fragment contained in the FastGrow database. The attachment point is represented as cyan stick.*

Caveats and Limitations The shape descriptors of the fragments or building blocks are pre-calculated on a conformer ensemble. When building your own FastGrow database (see Section 5.3), it might be necessary to increase the size of the generated conformer ensemble for the fragments or building blocks if these molecules are very flexible (i.e., contain a large number of rotatable bonds). Furthermore, the cylindrical shape descriptor (see Figure 1) has a fixed width (radius 10 Å) and a fixed length of 10 Å starting from the attachment point. These dimensions of the shape descriptor limit the size of the fragments or building blocks that can be stored in a FastGrow database. Furthermore, it is essential to note that the cylindrical shape descriptor for the binding pocket is calculated in the direction of the exit vector (see Figure 1A). Therefore, FastGrow is expected to deliver the best results if the binding pocket extends in this direction, i.e., the prospective fragments or building blocks screened from the FastGrow database are expected to be placed somewhere along the exit vector. On the other hand, it might be challenging for FastGrow to grow around the corner, i.e., to place fragments or building blocks into a binding pocket that extends in an orthogonal direction to the selected exit vector.

2 Technical Prerequisites

2.1 Required Software

This package of FastGrow is a command line application. Control through a graphical user interface (GUI) is available in SeeSAR, BioSolveIT's 3D flagship modeling platform. We highly recommend you also install SeeSAR to swiftly carry out all sorts of preparational steps, notably the selection of the growing position and direction (the exit vector) as well as any relevant pharmacophore definitions etc. The preparation can then be written to a `*.fastgrow` file (see also section 5.1) and used at the command line with FastGrow (this package).

Technically, you will need:

- The FastGrow **application package** (`https://biosolveit.de/download/?product=fastgrow`). Depending on your operating system, some libraries may have to be installed. Get in touch with us if that is the case:
  `mailto:support@biosolveit.com`. Please mention any errors/warnings that you see in your mail.
- A **shell** (Linux/Unix) or a terminal (macOS), or a command line environment (Windows; e.g.: cmd.exe or PowerShell)
- A valid **license** (from `mailto:license@biosolveit.de`), see below.

2.2 Licensing

FastGrow and FastGrowDBCreator need a license to operate that is available from us. There are various sorts of licenses, but in most of the cases your early testing will employ a license file that simply needs to be put next to the executable: Just drop the `*.lic` file that you received from us in the folder in which `fastgrow[.exe]` or `fastgrow_db_creator[.exe]` reside.

The license setup instructions will come with the license that we will send out — or that has already been sent out to you. In case you do not have a license yet, please get in touch with us at `mailto:contact@biosolveit.de`, and provide us with the necessary information. Please note: a valid SeeSAR license will be accepted by FastGrow, but will not work with the FastGrowDBCreator. You will need a separate license for FastGrowDBCreator.

License File Locations A "test license" that you can request online and that is sent to you instantaneously can simply be placed next to the executable (`fastgrow.exe`, `FastGrow` or `fastgrow` — depending on your operating system). For macOS please read on...

---

macOS Specialties On macOS, the executable will typically reside inside the *.app package:

```
/Applications/FastGrow.app/Contents/MacOS/FastGrow
```

To place the short term test license there, you will have to go into the *.app package using a right mouse click (or CTRL-click) on FastGrow.app in the Finder, and click on "Show package contents". In there, you will see the Contents subfolder, in there the MacOS subfolder, and in there, the FastGrow executable. If you are about to use the **test license**, place it right there, next to the executable. A longer term license will be handled separately, we will tell you how when we send that very license.

When you call FastGrow for the first time, go to the Finder, and navigate to the Applications folder. Do a right(!) click on FastGrow.app, and — if applicable — confirm that you want to open the program. It will flash up once, and you are good to go at the terminal prompt from there on.

Obtainig a License File Using `--license-info` you can obtain information about the specification of your license server machine, the searched directories, and the validity of the currently used license files. This may also be useful when FastGrow is not starting up as you would expect it to.

Call FastGrow with the `--license-info` option, to see an output like this:

```
.\fastgrow.exe --license-info
================================================================================
 License Information:
================================================================================
Host-ID: "00ff8c8dfdc6 0093376fa5e6 0293376fa5e5 0093376fa5e5 0093376fa5e9"
BIOSOLVE_LICENSE_FILE:
LM_LICENSE_FILE:

Currently used key/path after environment variables:
C:/Users/sajung/FastGrow-1.4.0

FastGrow:
 >> Valid key, expires on Tuesday, 31 December 2024



Request a license:
***
*** https://www.biosolveit.de/license/?product=fastgrow&operating_system=win64&...
***
```

Request an evaluation or longer-term license using the link that is provided at the very bottom of the output.

Also, this output may help us to find out if there are any problems with your license or its setup.

3 Jump Start: Structure-based Fragment Growing

To run a fragment growing with default options, you will need at least:

- A fragment database file (`.fastgrowdb`), either downloaded from the BioSolveIT website (`https://www.biosolveit.de/FastGrow/Libraries`) or generated with the FastGrowDBCreator tool (see Section 5.3)

(a) A protein and core ligand file

- A protein file (`.pdb` or `.ent` format)
- A core ligand file (`.sdf`, `.mol` or `.mol2` format), which is the molecule to grow from and which must have 3D coordinates, lie in the binding site of the specified protein and must have at least one pre-defined linker atom

(b) A fastgrow definition file

- A fastgrow definition file (`.fastgrow`) that you created with SeeSAR's Inspirator Mode.

Now run this — with the file names replaced by your own names of course:
In case (a), that is, if you provided a core ligand file and a protein file:

```
./fastgrow -d FragmentLibrary.fastgrowdb -p MyProtein.pdb -c MyCoreLigand.sdf
-o MyGrowingOutput.sdf
```

In case (b), that is, if you have prepared and exported a fastgrow definition file from SeeSAR:

```
./fastgrow -d FragmentLibrary.fastgrowdb -f MyDefinitionFile.fastgrow
-o MyGrowingOutput.sdf
```

Depending on the operating system that you use, please certainly also adapt the command line usage, e.g., use a slash or a backslash etc.

The above calls run FastGrow and create an output file `MyGrowingOutput.sdf` that, by default, contains a maximum of 100 growing solutions. More solutions can certainly be requested (see Section 4.2 on page 9). Additionally, it contains a score for every solution in an SD property field for post-processing purposes (`BIOSOLVEIT.FASTGROW_OPTIMIZED_SCORE` or `BIOSOLVEIT.FASTGROW_SCORE`, see Section 4.2).

4 Command Line Options

4.1 Overview

An overview of all commmand line options is available by calling FastGrow with `--help`. Default values are bracketed:

```
./fastgrow -h

 Program options:
-d [ --database ] arg             Fast grow database. Supported file type is: *.fastgrowdb
-p [ --protein ] arg              Protein file. Supported file types are: *.pdb, *.ent, *.cif, *.mcif and
                                  *.mmcif
-f [ --fastgrow-definition ] arg  Runs FastGrow on the basis of this FastGrow definition file (a .fastgrow
                                  file that can be exported from SeeSAR's Inspirator mode) - it contains
                                  the protein with the predefined binding site and core and linker.
                                  Note: Can't be used together with '--protein'.
                                  Supported file types: *.fastgrow
-c [ --core ] arg                 Core ligand molecule to grow.
                                  Note: The core ligand molecule must have 3D coordinates and lie in the
                                  pocket in the protein as it is also used to determine the binding site
                                  for growing.
                                  Supported file types: *.mol2, *.mol and *.sdf
-o [ --output ] arg               Molecule file with FastGrow results. Supported file type is: *.sdf
-n [ --nof-solutions ] arg (=100) The maximum number of solutions produced by FastGrow  (values between 1
                                  and 100000 are allowed).
-l [ --linker ] arg               The name of the linker atom in the input file. This argument is only
                                  needed if there are multiple linkers in the core.
--no-optimization [=arg(=1)]      Do NOT perform a restricted optimization to remove clashes and re-rank
                                  poses.
--no-clash-filter [=arg(=1)]      Do NOT perform filter step to remove severely clashing poses from output.
--no-smarts-filter [=arg(=1)]     Do NOT perform filter step to remove molecules containing reactive
                                  groups.

General options:
-h [ --help ]                     Print this help message.
--license-info                    Print license info.
--thread-count arg                Maximum number of threads used for calculations. The default is to use
                                  all available cores.
--version                         Print version info
-v [ --verbosity ] arg (=2)       Set verbosity level
                                       0 [quiet]
                                       1 [error]
                                       2 [warning]
                                       3 [info]
                                       4 [steps]
```

The abbreviated, one-letter options are preceded with one dash (-) whereas the longer, named options are preceded with two dashes (--). If an option needs an argument (arg), you can include or omit the equals sign. Adapt the command line usage to your operating system and shell.

4.2 Program Options

-d [--database] arg You must specify a FastGrow database file (`.fastgrowdb`) containing a library of prepared fragments or building blocks. You can download (`https://www.biosolveit.de/FastGrow/Libraries`) prepared FastGrow databases from our website covering up to 120,000 common building blocks and fragments. However, you can also generate your own FastGrow database using the FastGrowDBCreator tool (for more information, see section 5.3). The fragments contained in the FastGrow database file will be screened and suitable candidates will be attached to the core molecule that is defined via the `-c` option or contained in the FastGrow definition file given with the `-f` option.

Example:

```
fastgrow -d MyFragments.fastgrowdb
```

-p [--protein] arg Specify a protein file (`.pdb`, `.ent`, `.cif`, `.mcif` or `.mmcif`). If you do so, you have to make sure to also specify a core ligand with the `-c` option. The `-p` option is obsolete if you have prepared a FastGrow definition file (`-f` option).

Example:

```
fastgrow -p MyReceptor.cif
```

-c [--core] arg Specify the core ligand molecule, which is the molecule you want to grow from within the pocket. Supported file types are `.sdf`, `.mol` and `.mol2`. Please make sure the core ligand has 3D coordinates and lies in a pocket of the protein that you specified with the `-p` option. The core ligand must contain at least **one linker atom**, the bond to which defines the exit vector for the attachment of suitable building blocks (e.g. the growing direction) from the FastGrow database file. For more information, see Section 5.2. The core ligand is obsolete if you use a FastGrow definition file with the `-f` option (see below).

Example:

```
fastgrow -c MyCoreLigand.sdf
```

-f [--fastgrow-definition] arg Specify a FastGrow definition file (`.fastgrow`) which can be exported from SeeSAR's Inspirator Mode. The file contains all information required to run FastGrow, including the core ligand with defined exit vector, the binding site definition and optionally also pharmacophore constraints (see Section 5.1). Please note that this option cannot be used with the `-p` option. The separate specification of a protein (`-p`) and core ligand (`-c`) is obsolete if you have prepared a definition file.

Example:

```
fastgrow -f DefinitionFile.fastgrow
```

-o [--output] arg Takes a name for the output file which contains the FastGrow results (`.sdf` format). The output file will contain all top-scored growing solutions up to the maximum number of solutions defined with the `-n` option. By default, a maximum of 100 molecules is written (see `-n` option below). The output file contains a score for every solution in an SD property field for post-processing purposes (either `BIOSOLVEIT.FASTGROW_OPTIMIZED_SCORE` or `BIOSOLVEIT.FASTGROW_SCORE`, depending on whether the post-growing optimization has been done or not, see `--no-optimization` option below).

Example:

```
fastgrow -o MyGrowingResults.sdf
```

–n [––nof-solutions] arg(=100) Controls the maximum number of top-scored growing solutions that are written to the output file. The default is 100. Values between 1 and 100,000 are allowed.

Example:

```
fastgrow -n 2000
```

–l [––linker] arg Specify the identifier of the linker atom in the core ligand file you want to grow from. This is only needed if the core ligand (`-c` option) contains multiple linker atoms. You have to enter `R` followed by the atom ID of the linker atom. See Section 5.2 for additional information.

Example:

```
fastgrow --linker R1
```

––no-optimization Turns off the post-growing optimization step which re-ranks poses. The FastGrow default is to perform a pose optimization. You may want to skip the optimization step if the poses should be further processed with other tools. Please note: If this option is specified, the annotated score in the output file is the `BIOSOLVEIT.FASTGROW_SCORE`.

Example:

```
fastgrow --no-optimization
```

––no-clash-filter Turns off a post filter which removes molecules with intolerable clashes form the FastGrow solutions. The FastGrow default is an active clash filter. You may want to leave initial clashes in case that `--no-optimization` is set and the poses should be processed further.

Example:

```
fastgrow --no-clash-filter
```

––no-smarts-filter Turns off a SMARTS-based post-filter which removes molecules with reactive or unwanted groups. Such groups may form during the fragment growing step or be present in the attached building block. The SMARTS rules are hard coded and inaccessible to the user. They comprise only highly reactive species such as, for example, peroxo-groups.

Example:

```
fastgrow --no-smarts-filter
```

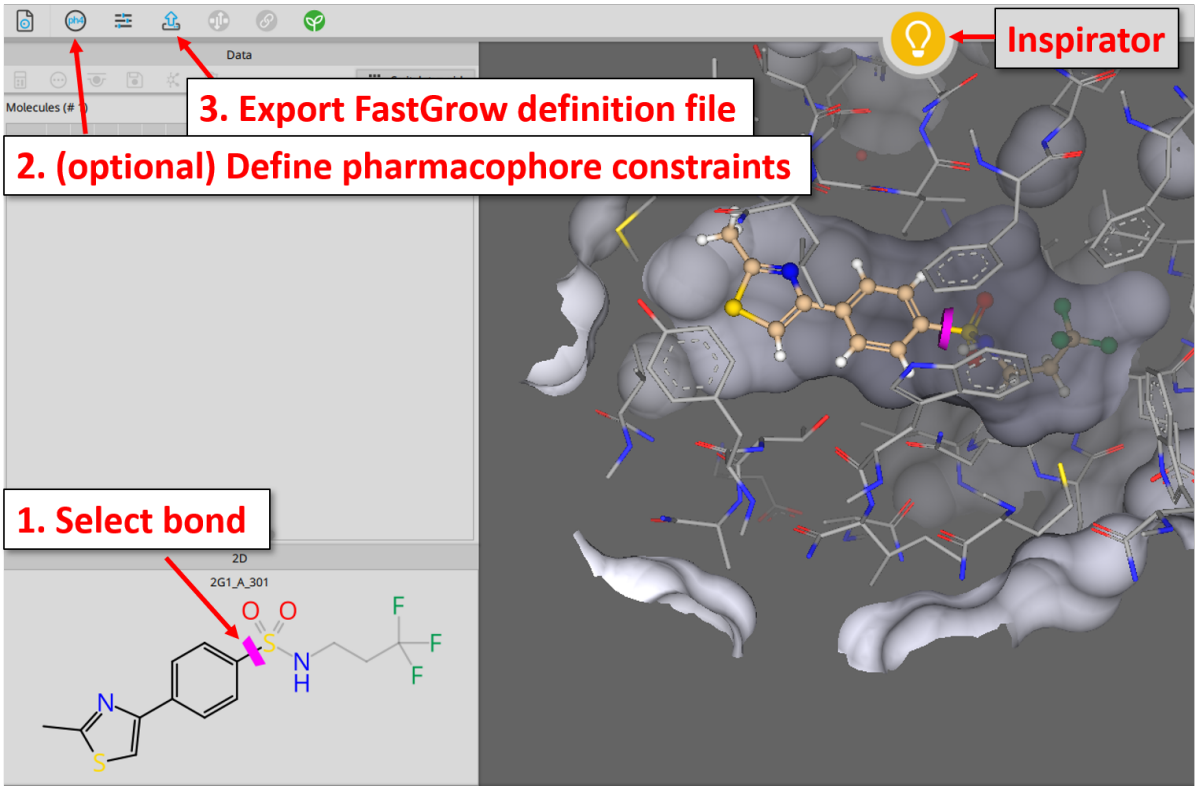

Figure 2: *Generate a FastGrow definition file in SeeSAR's Inspirator Mode. Step 1: click on the appropriate bond to define the exit vector. Step 2: define your pharmacophores with the ph4 button (optional). Step 3: export the FastGrow definition file.*

5 Generating Input Files for FastGrow

5.1 Preparing a FastGrow definition file

The recommended way to run FastGrow at the command line is by providing a FastGrow definition file which can be conveniently prepared in SeeSAR. To get an idea on how to prepare such a file, you can download a co-crystallized structure in SeeSAR, e.g., PDB 4M3G, extract the ligand and then transfer it to the Inspirator Mode. Next, click on the appropriate bond (either in 2D or the 3D window) to define the exit vector (see Figure 2). The part of the molecule to be replaced is then grayed out. If you want the other part of the molecule to be replaced, simply click again on the bond to revert the direction of growing. Now, you can conveniently export the definition file and use it with the FastGrow command line tool (see Figure 2).

It is also possible to include sphere pharmacophore constraints (e.g donor and acceptor interactions, hydrophobic contacts etc.) into the fastgrow definition file. These pharmacophores can also be defined conveniently in SeeSAR's Inspirator Mode (via the ph4 button, see Figure 2). FastGrow uses these constraints to filter suitable shape-matching poses for the specified pharmacophores. This will constrain the algorithm to deliver only compliant growing solutions.

5.2 Preparing a core ligand file

An alternative way of running FastGrow is to provide a protein and a separate core ligand file containing a linker atom. This is obsolete if you exported a FastGrow definition file (see Section 5.1). The following

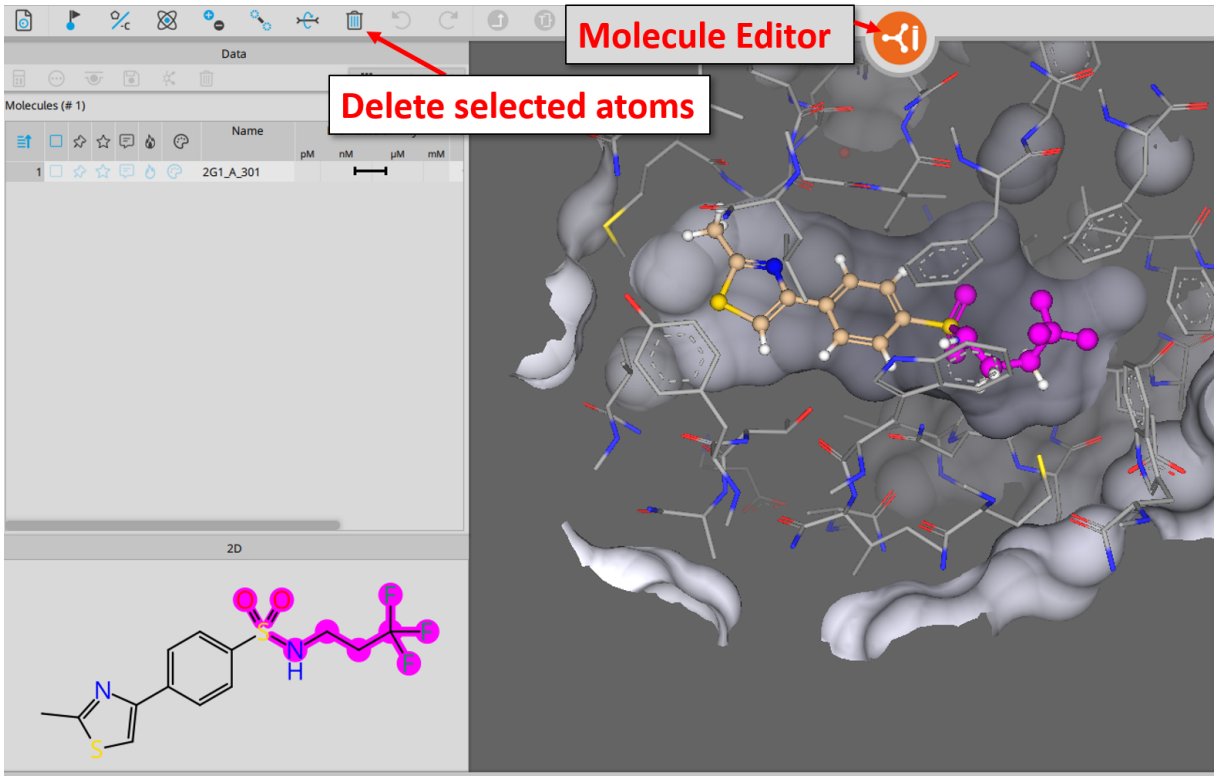

Figure 3: *Delete a part of the binding site ligand (pink selection) to generate a core ligand.*

example demonstrates how to prepare a core ligand file manually. To follow along, download a protein file containing a ligand in SeeSAR. In our example this is 4M3G (see Figure 2). Extract the ligand and transfer it to the Molecule Editor Mode. Within the Molecule Editor mode, you can now delete a part of the ligand by selecting the appropriate atoms (either in 2D or 3D window) and clicking on the waste bin icon (as shown in Figure 3). In the remaining core ligand fragment you can now select one atom to be replaced by a linker (in this example the sulfur atom of the remaining SH group). Click on the appropriate atom and then choose `[R]` within the "Change element" tab (see Figure 4). The core ligand prepared in that way can now be saved to the Molecules Table and then exported as an SD file. This file will then be given to the FastGrow command line tool using the `-c` option. Make sure you also specify the respective protein file (4M3G.pdb in this example) with the `-p` option.

It is also possible to define multiple linker atoms in a single core ligand and use it with FastGrow, for example if you want to grow into two different directions of the binding site. However, FastGrow can use only one exit vector in a single run. In this case, you must specify the linker atom you want to grow from in the actual run. This is done with the `-l / --linker` option by entering `R` followed by the respective atom ID, e.g. `-l R1` or `-l R13` for the example in Figure 5. You can visualize the atom ID in SeeSAR, e.g., by pressing and holding the L key on your keyboard and then clicking on the linker atoms. Alternatively, you can open the SD file containing the core ligand with a text editor and manually extract the atom ID of the linker atoms (labeled R or R# in the SD file).

5.3 Preparing a FastGrow database file

To use your own fragments for growing with FastGrow, you must prepare a FastGrow database file (`.fastgrowdb`) with the FastGrowDBCreator tool contained in this package. You need a separate license to run FastGrowDBCreator (`mailto:license@biosolveit.de`). In case you have a testing or short term license from us, please put it next to the executable (`fastgrowdbcreator[.exe]`) in the same way as described for FastGrow in Section 2.2.

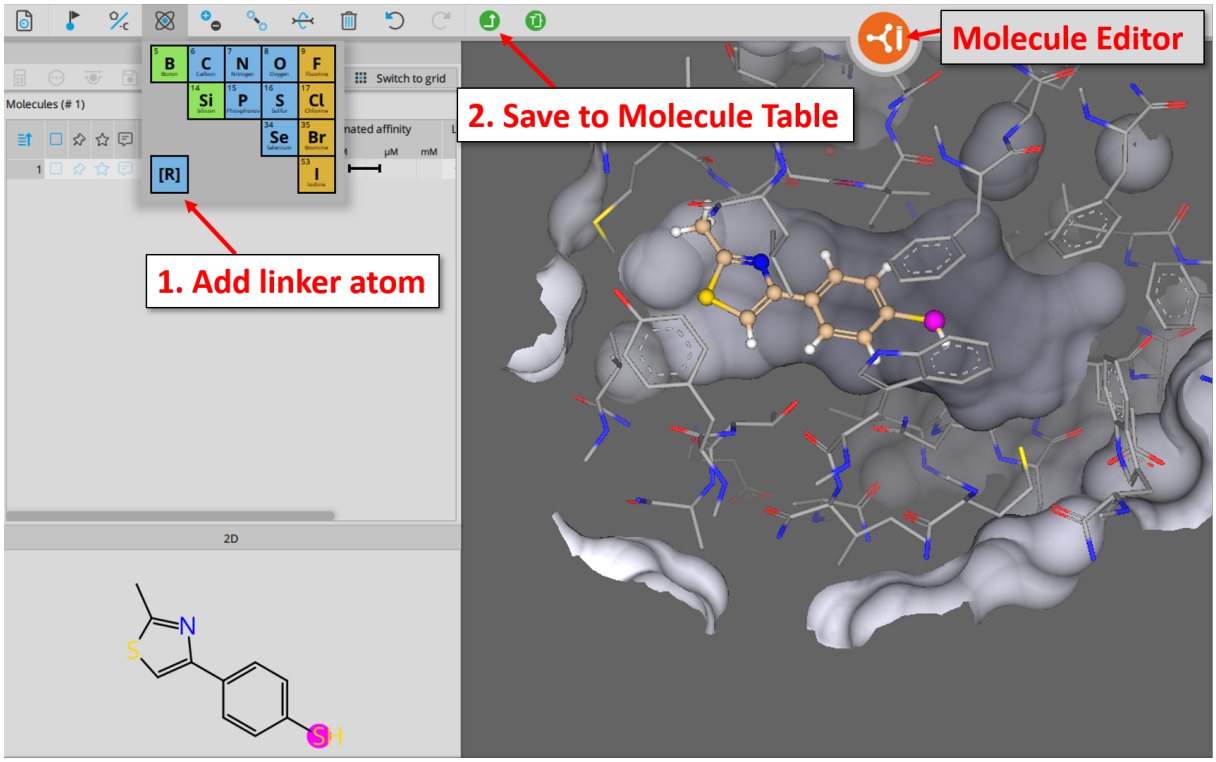

Figure 4: *Step 1: Add a linker atom replacing the exocyclic sulfur (pink) to the core molecule. Step 2: Save it to the Molecules Table within SeeSAR's Molecule Editor Mode.*

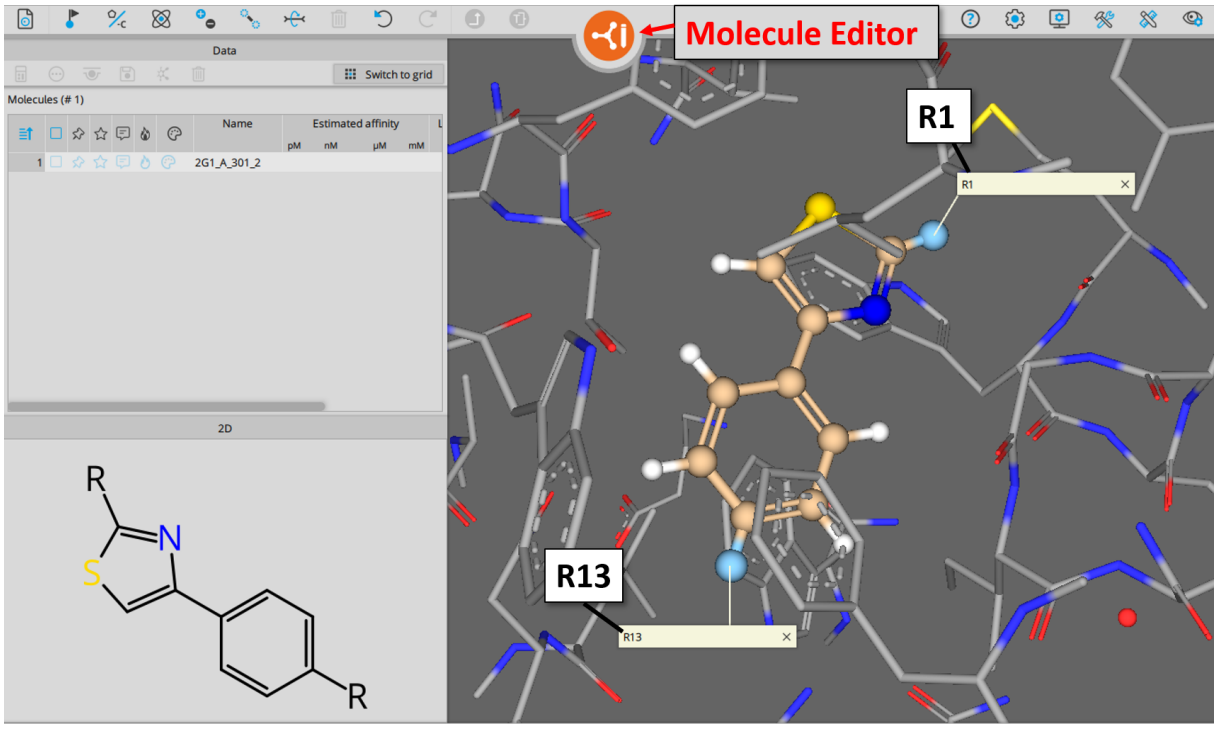

Figure 5: *Addition of a second linker atom replacing the methyl group at the thiazole ring to the core ligand used as example before (see Figure 4). You can visualize the atom IDs of the linker atoms by pressing and holding the L key on your keyboard and then clicking on the linker atoms.*

As already outlined in the Introduction (Section 1), specific cylindric RVM descriptors have to be pre-computed for every fragment you want to use with FastGrow. This is done with the FastGrowDBCreator tool. You can also download (`https://www.biosolveit.de/FastGrow/Libraries`) prepared FastGrow databases containing up to 120,000 compounds from the BioSolveIT website. Every fragment molecule passed to the FastGrowDBCreator must have exactly one linker atom to be processed (see Figure 4 for an example of manually adding a linker atom in SeeSAR). Please get in touch with us and we will help you setting up an automated workflow for this task. Once you have generated a suitable input file, you can use it with the FastGrowDBCreator tool which will then automatically perform all needed steps to compute the conformers and shape-based descriptors for every fragment and stores them in the output FastGrow database.

5.3.1 FastGrowDBCreator: Help

An overview of all command line options is available by calling FastGrowDBCreator with `--help`. Default values are bracketed:

```
./fastgrow_db_creator -h

 Program options:
-i [ --input ] arg                  Fragments to add to the database. Fragment molecules must have exactly
                                    one linker atom. Supported file types are: *.smi, *.smiles, *.mol2,
                                    *.mol and *.sdf
-o [ --output ] arg                 Output FastGrow database. Supported file type is: *.fastgrowdb
-n [ --nof-conformations ] arg (=10) Number of conformations to generate for the fragments.
--no-smarts-filter [=arg(=1)]       Do NOT perform filter step to remove molecules containing reactive
                                    groups.
-n [ --mode ] arg (=0)              Execution mode, allowed modes are:
                                         0 [create (create a new database)]
                                         1 [overwrite (overwrite existing database)]
                                         2 [append (add molecules to an existing database)]
--error-output [=arg(=1)]           Write detailed error output files containing molecules not added to
                                    the output database.

General options:
-h [ --help ]                       Print this help message.
--license-info                      Print license info.
--thread-count arg                  Maximum number of threads used for calculations. The default is to use
                                    all available cores.
--version                           Print version info
-v [ --verbosity ] arg (=2)         Set verbosity level
                                         0 [quiet]
                                         1 [error]
                                         2 [warning]
                                         3 [info]
                                         4 [steps]
```

5.3.2 FastGrowDBCreator: Program Options

–i [–-input] arg Specify a library of fragments that you want to incorporate into a FastGrow database. Supported file types are `.smi`, `.smiles`, `.mol`, `.mol2` and `.sdf`. All fragment molecules must contain **exactly one linker atom**, otherwise they will be skipped and not added to the database. You can manually add a linker atom to compounds in SeeSAR's Molecule Editor Mode (see Figure 4). Get in touch with us and we will help you setting up an automated workflow for this task.

Example:

```
fastgrowdbcreator -i MyFragmentsWithLinker.sdf
```

–o [–-output] arg Specify the output file (`.fastgrowdb`) to which the prepared fragments should be added. You can select from different modes regarding how the output file is written (see `-m` option below).

Example:

```
fastgrowdbcreator -o MyDatabase.fastgrowdb
```

–n [––nof-conformations] arg(=10) The maximum number of conformers generated for each fragment contained in the input file. The default value is a maximum of 10 conformers.

Example:

```
fastgrowdbcreator -n 100
```

––no-smarts-filter By default, a SMARTS-based filter step is performed to remove fragments from the input file which contain reactive groups. You can switch off the filter step with this option. The SMARTS rules are hard coded and inaccessible to the user. They comprise only highly reactive species such as, for example, peroxo-groups.

Example:

```
fastgrowdbcreator --no-smarts-filter
```

–m [––mode] arg(=create) You can select from three different "execution modes" that steer how the input fragments affect the FastGrow output database. The default mode is to *create* a new database file. However, you can also choose to *append* new fragments to an existing FastGrow database: This may be very convenient in cases where you want to gradually build up a large database. The following modes can be selected:

- `create` Generate a new FastGrow database containing the processed fragments from the input file.
- `overwrite` Overwrite an existing FastGrow database with the processed fragments from the input file.
- `append` Add all successfully processed fragments from the input file to an existing FastGrow database.

Example:

```
fastgrowdbcreator -m append
```

––error-output With this option used, you will generate a detailed error output file for every fragment contained in the input file whenever it was not properly processed and could not be added to the FastGrow database.

Example:

```
fastgrowdbcreator --error-output
```

6 General Options

The general options described in this chapter are valid for both FastGrow and FastGrowDBCreator.

–h [––help] Displays the command line help with short descriptions for every argument option. For more information see Sections 4.1 and 5.3.1.

Example:

```
fastgrow ––help
```

––license-info Shows command line information about the license setup you currently use. If you have any problems with your license, send an email to `mailto:support@biosolveit.com` and include this information. For more information see Section 2.2.

Example:

```
fastgrow ––license-info
```

––thread-count arg Specify the maximum number of threads used. By default, all available logical cores of your computer are used. You may want to reduce the number of threads used if you want to run other computations on your computer at the same time, or if you share the compute resource.

Example:

```
fastgrow ––thread-count 4
```

––version Displays information on the version of FastGrow or FastGrowDBCreator on the command line. In quoting the respective tool, please mention this version number.

Example:

```
fastgrow ––version
```

–v [––verbosity] arg(=2) Set the verbosity level, e.g., the level of console output, with an integer argument. The default value is 2. The following options are available:

- `0` Quiet. No messages will be displayed on the console. Errors will be ignored whenever possible.
- `1` Error. Only error messages will be displayed.
- `2` Warning. The default setting, warnings and error messages will be displayed.
- `3` Info. Additional information beyond errors and warnings are displayed.
- `4` Steps. In addition to the 'Info' option, the progress is displayed in detail.

Example:

```
fastgrow –v 0
```

7 Further Reading, References

The basic ideas of the shape-based descriptor and algorithm behind FastGrow are covered in the original publication by Patrick Penner.[1] More information on the tool is available at

`https://biosolveit.de/FastGrow`.

Complementary tools, especially also the graphical platforms SeeSAR and infiniSee, can be obtained from the BioSolveIT website (`https://biosolveit.com`).

References

[1] Patrick Penner, Virginie Martiny, Arnaud Gohier, Marcus Gastreich, Pierre Ducrot, David Brown, and Matthias Rarey. Shape-based descriptors for efficient structure-based fragment growing. *Journal of Chemical Information and Modeling*, 60:6269–6281, 12 2020.

We wish you great success and much joy with FastGrow!