User Guide for version 1.0

How to cite?

- Filipovic, J., Vavra, O., Plhak, J., Bednar, D., Marques, S., Brezovsky, J., Matyska, L., Damborsky, J., A Novel Method for Analysis of Ligand Binding and Unbinding Based on Molecular Docking. (in preparation).
- Vavra, O., Filipovic, J., Plhak, J., Bednar, D., Marques, S., Brezovsky, J., Matyska, L., Damborsky, J., CAVERDOCK: A New Tool for Analysis of Ligand Binding and Unbinding Based on Molecular Docking. PLOS Computational Biology (in preparation).

1 Introduction

CaverDock is a tool designed for a rapid analysis of transport processes in proteins [2, 3]. It models the transportation of a ligand – a substrate, a product, an inhibitor, a co-factor or a co-solvent – from the outside environment into the protein active or binding site (or *vice versa*).

The input for the calculation is a protein structure in the PDB format and a ligand structure in the PDBQ format. The outputs are a ligand's trajectory and energetic profile. CaverDock implements a novel algorithm which is based on molecular docking and is able to produce a contiguous ligand trajectory and estimation of the binding energy along the pathway.

The current version of CaverDock uses Caver [4] for the pathway identification and a heavily modified Autodock Vina [1] as the docking engine. The tool is much faster than molecular dynamic simulations (usually 2-20 min per job), making it suitable even for virtual screening. The software is extremely easy to use, since in its minimalistic configuration it requires only the setup for AutoDock Vina and the geometry of the tunnel.

2 Installation

In current version, CaverDock is distributed as pre-compiled binaries only. There are several tools and libraries that need to be installed in the system:

- Python 2.7 (3.x is currently not compatible with vPython)
- SciPy
- vPython
- docopt
- CGAL
- boost
- MPI
- MGLTools

On Ubuntu 16.04, dependencies can be installed by as follows.

Listing 1: Dependencies installation

```
sudo apt-get install libopenmpi-dev libboost-all-dev \
  python-visual python-docopt libcgal-dev python-scipy \
  autodocktools
```

CaverDock archive contains several folders:

- `bin`, where the binary of caver-dock is placed;
- `discretizer`, containing sources of the discretizer;
- `scripts`, where several scripts for a more comfortable CaverDock usage are located;
- `doc` contains this documentation;
- `example` containing a simple example to demonstrate the CaverDock workflow described in the Section 3.

In the root folder, there is a script setting all paths. The last step of the installation is to execute this script, so CaverDock and additional scripts can be called without using an absolute path.

Listing 2: Path setup

```
source caverdock.bashrc
```

3 Quick start

The typical CaverDock workflow consists of the following steps:

- Calculation of protein's tunnels with CAVER. Selection of tunnel(s) for CaverDock analysis and their export into a PDB file.
- Discretization of tunnel(s) to a set of discs with CaverDock's tool Discretizer.
- Preparation of input files for docking (by conversion of PDB to PDBQT format, optionally setting the side-chain flexibility).
- Configuration and execution of CaverDock.
- Visualization of results from CaverDock computation.

In the following text, we will describe the individual steps of the workflow. Each reader may test the workflow mentioned in this section using the example input stored in the `example` folder packed with CaverDock. Any step may be omitted as the folder contains also intermediate and final results.

3.1 Tunnels calculation

The creation and selection of tunnels can be done locally by CAVER software or using the web portal [5]. For more details on CAVER usage, see CAVER User Guide [6]. You can find the already exported tunnels in the example packed with the CaverDock tool.

3.2 Tunnels discretization

Having one or more tunnels exported, they must be discretized to set of cuts (discs evenly cutting the tunnel). The discretizer tool requires a tunnel in PDB format as the input and produces a discretized tunnel for CaverDock. It is recommended to switch visualization on (option `-d`) and check if the tunnel is discretized correctly (a visualized tunnel can be rotated by mouse movement when the right mouse button is pressed). See Listing 3 for an example of the discretizer execution and Figure 1 for an illustration of visual output.

Listing 3: Discretizer execution

```
discretizer.py -f tunnel1.pdb -d -o tunnel1.dsd
```

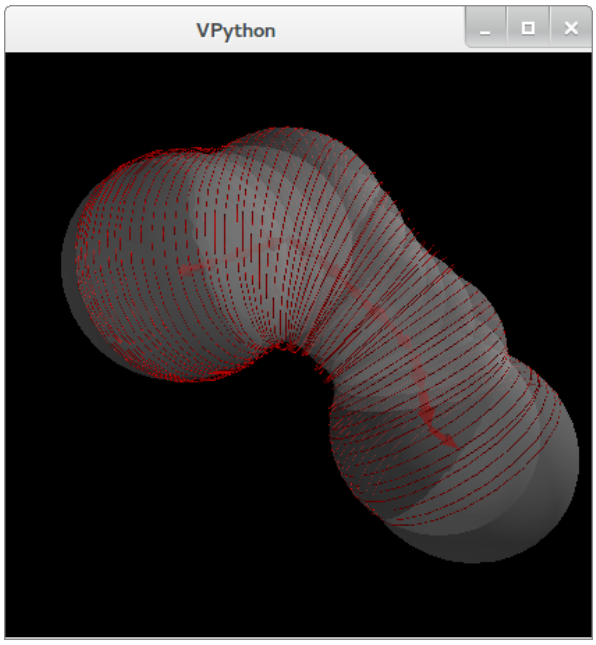

Figure 1: Visualization of the tunnel discretization.

3.3 Molecules preparation

The molecules of the receptor and the ligand must be prepared for molecular docking. The procedure is the same as with Autodock Vina – scripts in MGLTools [7] can be used for this purpose, as shown in Listing 4.

Listing 4: Molecules preparation

```
prepare_receptor4.py -r receptor.pdb
prepare_ligand4.py -l ligand.mol2
```

3.4 CaverDock execution

Finally, CaverDock must be configured and executed. The minimalistic configuration must contain a specification of the search box and names of a receptor, a ligand and a tunnel file. Please note that the search box must contain the whole ligand's trajectory (i.e. the whole tunnel) together with the area of the possible movement of side-chains. Although the configuration file can be prepared manually or with MGLTools, CaverDock package contains a script which assembles the configuration file with basic settings automatically.

Listing 5: Configuration preparation

```
prepareconf.py -r receptor.pdbqt -l ligand.pdbqt \
-t tunnel1.dsd > caverdock.conf
```

With the prepared configuration file, CaverDock may be executed. CaverDock uses MPI[1] for parallel execution. For tunnel analysis, it must be executed in at least two processes (by setting `mpirun` parameter `-np` to 2 or more).

Listing 6: CaverDock execution

```
mpirun -np 9 caverdock --config caverdock.conf --out test
```

3.5 Results visualization

Files test-lb.pdbqt and test-ub.pdbqt are created upon a successful processing of the tunnel by CaverDock. The first file contains a trajectory estimating the lower-bound of energy profile. The lower-bound trajectory is not contiguous, however, it samples the tunnel without any gap in the ligand movement. The second trajectory is contiguous and estimates the upper-bound of the energy profile (it is the best trajectory found, however, a trajectory with better energy profile may exist). We can explore the geometry of trajectories by standard tools allowing to view PDBQT files (PMV and some versions of PyMOL) and we can also create graphs of transport energy.

Energies of all snapshots as well as their positions in a tunnel are stored in PDBQT files (lines beginning with REMARK CAVERDOCK). One can easily extract the data to text with a simple script.

Listing 7: Extraction of energies

```
energyprofile.py -d tunnel1.dsd  -t test-ub.pdbqt -s 0 > energy.dat
```

The energies should be extracted from PDBQT file containing an upper-bound trajectory (it contains geometry of upper-bound trajectory and both upper and lower-bound energies). After the extraction, one can easily create graphs with any tool, such as gnuplot.

Listing 8: Visualization of energies with gnuplot

```
gnuplot -e "set xlabel \"distance\"; set ylabel \"energy\"; plot "\
"\"energy.dat\" u 1:4 w l t \"upper-bound\", \"\" u 1:6 w l "\
"t \"lower-bound\"" -p
```

[1] Implementation of Message Passing Interface, such as OpenMPI: https://www.open-mpi.org/

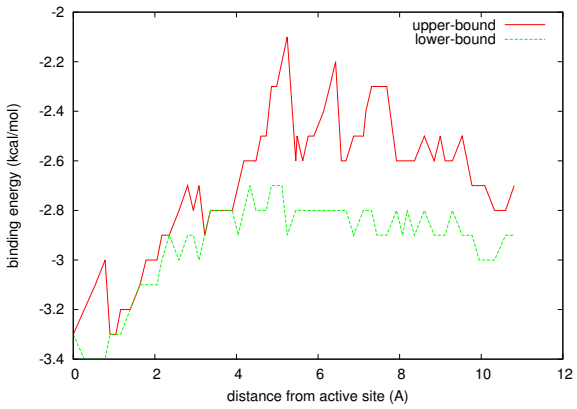

Figure 2: Visualization of energy barriers along the trajectory. The upper-bound trajectory is shown in red, the lower-bound trajectory is shown in green.

4 Results interpretation

To fully understand CaverDock results, one needs to be familiar with basic principles of the CaverDock's method, which is used to analyze a ligand's movement through a tunnel. The method uses constrained docking, with two types of constraints:

- placement of a ligand atom at 2D disc in 3D space;
- placement of the whole ligand in upper-bound vicinity of defined snapshot.

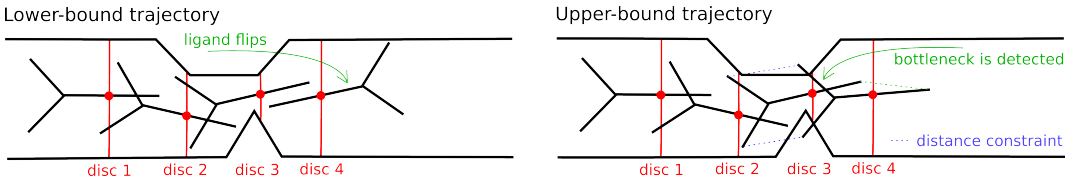

Figure 3: A scheme depicting CaverDock method providing the lower-bound (left) and the upper-bound (right) trajectories.

First, CaverDock computes the so called *lower-bound trajectory*. With this trajectory, a ligand atom (closest to the ligand center by default) is iteratively placed and consequent discs. This method samples the tunnel finely, however, a ligand trajectory may contain discontinuities (a ligand can e.g. flip when moved to a different disc). Thus, some bottlenecks may not be detected (Figure 3 left). Second, CaverDock starts to search for a contiguous trajectory by docking to consequent discs, but always in a vicinity of the previous snapshot (Figure 3 right).

Snapshots of the lower-bound trajectory are computed for each disc separately, thus, the trajectory search space is relatively small. On the other hand, the position of each snapshot of the contiguous trajectory depends on the previous snapshot. As there are exponentially many trajectories with respect to the number of discs, an exhaustive search of all contiguous trajectories is not feasible. CaverDock uses the heuristic method to prune the space of possible

trajectories. The heuristics does not guarantee that the trajectory with the lowest energy is found, thus, we call a contiguous trajectory *upper-bound trajectory*, as it upper-bound energy of transport process (the actual energy is the same or lower).

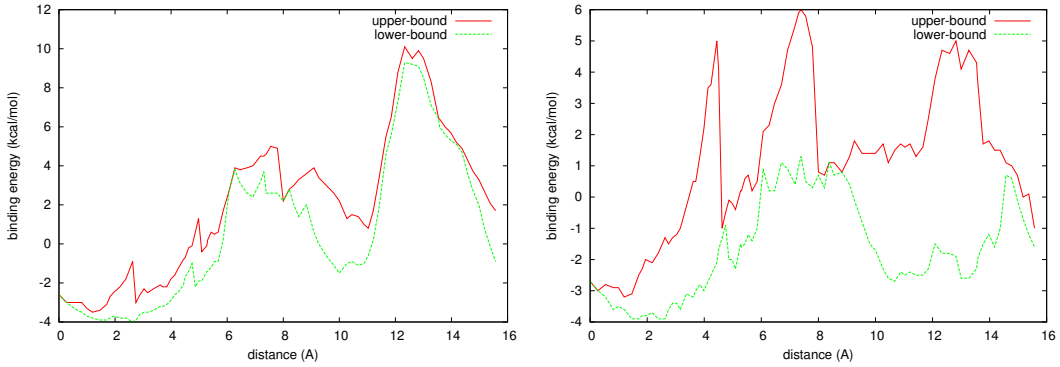

Figure 4: Examples of two distinct trajectories obtained using CaverDock. Left: similar lower and upper-bound, right: different lower and upper-bound.

Ideally, the lower-bound and upper-bound energies approximately agree (Figure 4 left). If the difference is higher than expected(Figure 4 right), we need to explore the trajectory and examine whether the bottleneck which appeared in the upper-bound trajectory is rather an artifact of the computational method, or if it is created by a real barrier in the tunnel. We can also try to improve the precision of the CaverDock computation using the procedure described in the Section 6.

5 Command line options

The main function of CaverDock is to analyze the transportation of a ligand through a tunnel. However, it can be executed also for single docking. In the single docking mode, CaverDock allows to perform single docking with constraints, which can be used as a building block for other tools using constrained docking.

The parameters accepted by CaverDock can be divided into three categories:

- docking options, derived from AutoDock Vina;
- options configuring tunnel analysis;
- options for constrained docking.

5.1 Docking options

CaverDock accepts the same parameters as Autodock Vina, which are used to configure, how the docking is performed. When only those parameters are given and only one process is executed[2], CaverDock works exactly like AutoDock Vina. When a tunnel analysis or a single constrained docking is executed, docking options described in this section must also be defined, as they parametrize docking internally used in CaverDock. The list of basic docking

[2]by a direct execution of the binary, or executing it via `mpirun -np 1`

parameters can be found in Table 1. For more information about AutoDock Vina docking and all docking parameters, we refer to AutoDock Vina Users Guide [8].

`--receptor arg`	Rigid part of the receptor (in PDBQT format).
`--flex arg`	Flexible side chains, if any (in PDBQT format).
`--ligand arg`	Ligand (in PDBQT format).

`--center_x arg`
`--center_y arg`
`--center_z arg`
	Coordinates of the center of the search box. The search box must contain space of the whole tunnel and all flexible residues with their potential movement. The CaverDock package contains a script `prepareconf.py` which computes the search box automatically (Section 3.4).

`--size_x arg`
`--size_y arg`
`--size_z arg`
	Size of the search box in Å. The search box must contain space of the whole tunnel and all flexible residues with their potential movement. The CaverDock package contains a script `prepareconf.py` which computes the search box automatically (Section 3.4).

`--out arg`	Output file(s) prefix. When docked to the tunnel, a file with the suffix `-lb.pdbqt` is generated for the lower-bound trajectory, `-up.pdbqt` for the upper-bound trajectory and occasionally `-failed.pdbqt` if the upper-bound trajectory is not computed (the failed trajectory contains the longest known contiguous trajectory).
`--exhaustiveness arg`	Exhaustiveness of the global optimization algorithm (a higher number means a higher number of tested conformations). Usually values from the interval 1–8 are sufficient.
`--cpu arg`	Number of threads used per process.
`--log`	Writes a log file(s), usable for debugging.

Table 1: Docking parameters.

5.2 Tunnel analysis options

In this section, we introduce all parameters related to the analysis of a ligand transport through a tunnel. For the transport analysis, at least two processes must be executed (using `mpirun --np 2` or greater). Note that the only required parameter is `--tunnel`, other parameters can be used to change the default settings. The basic parameters allow users to define the type of the computed trajectory and the optimization strategy or to hint the initial position of a ligand (Table 2), whereas advanced parameters affect constraints applied to the docking (Table 3).

`--tunnel arg`	The file containing discretized tunnel. This parameter is required.
`--final_state arg`	The final state of trajectory search. May be set to LB (compute the lower-bound trajectory only) or SMOOTHED (default), computing the lower and the upper-bound trajectories.

`--optimization_strategy arg`	Strategy for the upper-bound trajectory optimization. CaverDock may minimize the highest energy in a trajectory (MAX, default value) or the integral of energy across a whole trajectory (INTEGRAL).
`--dock_like arg`	PDBQT file defining ideal position of the ligand in the active site. The starting disc is selected to bring a ligand closest to the active site. This parameter may help when the tunnel obtained from CAVER is too deep (and thus CaverDock pushes the ligand against the energetic barrier at the bottom of the tunnel).
`--dump_bottlenecks arg`	Dump the bottlenecks observed along the trajectory. There are multiple files generated: in the file `arg`, a list of bottleneck residues and a list of bottlenecks for final upper-bound trajectory are given. For n disks, files `arg.0`.., `arg.n-1` are generated: they contain the bottlenecks for the snapshots used in the final trajectory as well as the snapshots generated during the trajectory search and not used in the output trajectory. Reporting the bottlenecks in unused snapshots may help the user to recognize which residues prevent CaverDock from moving the ligand in alternative pathways.

Table 2: Basic tunnel analysis parameters.

Whereas the basic tunnel analysis options can be set by beginner users and should not affect the robustness of the CaverDock computation, a suboptimal setting of advanced options may lead to very a slow computation or a failure of the search for contiguous conformation. Thus, these should be set carefully.

`--backtrack_threshold arg`	The energy increment over the lower-bound trajectory, which leads to the execution of backtracking. A lower value, a lower energy of the upper-bound trajectory and higher computation time can be expected. The default value is 1 kcal/mol.
`--backtrack_limit arg`	The minimal number of steps (visited discs) which must be performed after backtracking, before a new backtracking is executed. Values higher than 1 forbid frequent executions of backtracking from unfeasible areas. The default value is 5.
`--cont_threshold arg`	The maximal atom movement (in Angstroms) between adjacent snapshots, which is considered to be contiguous. The default value is 0.8 Å.

`--pattern_limit arg`	The maximal distance (in Angstroms) between adjacent snapshots, which is not penalized by a contiguity constraint. The default value is 0.6 Å, must be smaller than `--cont_threshold` and higher than the maximal distance of discs created during a tunnel discretization.
`--allow_flex_discontinuity`	Allows the receptor side-chain residues to perform non-contiguous movements even when the upper-bound trajectory is computed. The default behavior is that a contiguous movement is required for both the ligand and the side-chain residues.
`--parallel_workers_lb arg`	The number of processes solving each docking task of a lower-bound trajectory computation in parallel (alternative to exhaustiveness with better scaling). The default value is 4.
`--parallel_workers_smooth arg`	The number of processes solving each docking task of an upper-bound trajectory computation in parallel (alternative to exhaustiveness, scales better). The default value is 4.

Table 3: Advanced tunnel analysis parameters.

5.3 Constrained docking options

In this section, parameters applicable to perform a single docking with constraints are introduced. Note that those parameters are not employed in the standard usage scenario of CaverDock (an analysis the whole trajectory). However, they may be used for debugging or building a tool on the top of CaverDock.

There are essentially two types of constraints implemented: a disc, which fixes a ligand's atom and a pattern, which restricts the positions of all ligand's atoms. When the disc is defined, docking is performed in the way that a selected ligand atom must be placed in a close vicinity of the disc. The discs may define a cut of a tunnel, or a point in space (when the disc radius is set to a very small number). The pattern holds the whole ligand in a vicinity of pattern atoms. So, it is possible to search only for the ligand conformations which are close to the one defined by the pattern.

`--ccenter_x arg`
`--ccenter_y arg`
`--ccenter_z arg`
	Coordinates of the center of a disc attracting a ligand (its central atom, or the atom selected by `catomnum` parameter).

`--cnormal_x arg`
`--cnormal_y arg`
`--cnormal_z arg`
	The normal vector of a disc attracting a ligand (its central atom, or the atom selected by `catomnum` parameter).

`--cradius arg`	The radius of a disc attracting a ligand (its central atom, or atom selected by `catomnum` parameter).
`--catomnum arg`	The ID of an atom in PDB, which is attracted to a disc.

`--ctemplate arg`	PDB or PDBID of the template, which restricts the ligand atoms movement (they must remain in a vicinity of the template atoms, defined by `--ctemplate_limit`).
`--ctemplate_limit arg`	The distance (in Angstroms) of the docked ligand atoms to their counterparts in a template (defined by –ctemplate), which is tolerated without applying an attractive force to the template.

Table 4: Constrained docking parameters.

6 Best practices

6.1 Improving energy

CaverDock has been parametrized to bring a good mix of computational efficiency and precision of computation. However, there is still some room for hand-tuning the CaverDock computation. One can tune the CaverDock parameters to increase the precision of its computation and the structure of molecules to ease the computation of the ligand movement.

6.2 Improving lower-bound energy

The lower-bound energy depends on a tunnel geometry and the docking ability to find a good local minima. To improve (decrease) the energy of the lower-bound trajectory, several arguments can be tuned:

- `--exhaustiveness` can be set to a higher value which increases the number of random walks of Markov-chain Monte Carlo global search algorithm and increases the probability of finding a good local minima;
- `--parallel_workers_lb` can be set to a higher value and should have similar effect to `--exhaustiveness`.

If the energy of the lower-bound trajectory is too high even with high exhaustiveness, it is very probable that the ligand cannot pass through the tunnel in thereal-world, or there is some issue with the tunnel or the receptor geometry. Three typical issues are described below.

- The tunnel obtained from CAVER forces the ligand to move too deeply into the tunnel, where it reaches the energetic barrier at the tunnel bottom. This situation can be detected by exploring the energy graph: the energy is very high in the area of the beginning of the tunnel (i.e. around the position zero at x-axis). In such case, `--dock_like` parameter can be used to set the correct starting disc, so the ligand will not be pushed against the tunnel bottom.
- There are side-chain residues forming a bottleneck. We can detect this issue by exploring the bottleneck dump. In such case, side-chain residues forming the bottleneck should be set to be flexible by MGTools [7].
- The backbone residues are forming a bottleneck. We can detect this issue by exploring the bottleneck dump. In such case, we need to use a different geometry of the receptor

(e.g. one obtained from an MD simulation). This is a typical issue when the receptor structure is taken from a crystal with tunnels closed due to intramolecular interactions and crystal packing.

6.3 Improving upper-bound energy

When we are satisfied with the lower-bound energy, we can focus on the upper-bound energy. If the difference between the lower-bound and the upper-bound energies is too high, we can tune several CaverDock parameters:

- `--backtrack_threshold` can be set to a lower value, if we consider 1 kcal/mol already as an undesired difference between the lower-bound and the upper-bound energies. Beware that tuning this value may result in longer computation times.
- `--backtrack_limit` can be set to a lower value, so CaverDock will execute backtracking more aggressively. Beware that tuning this value may result in longer computation times.

Together with the CaverDock parameters, we should also check the geometry of the tunnels and the receptor (Section 6.2), as some bottlenecks as well as issues with a tunnel bottom may arise only when a contiguous trajectory is computed.

6.4 Improving computation time

The CaverDock is usually quite fast on a standard desktop computer (its execution commonly takes from minutes to dozens of minutes). However, the execution time can be improved by the following actions.

- Use as low flexible side-chain residues as possible. The computational time of a docking grows rapidly with the number of degrees of freedom of the system. Using flexible side-chain residues may greatly improve the energy profile, however we recommend setting flexibility only on residues, which are proved to form a bottleneck in a particular tunnel (i.e. they are reported with `--dump_bottlenecks` option).
- Use the right number of parallel processes. CaverDock uses MPI, thus, it can use multiple cores of a desktop machine or even multiple nodes of a cluster. The number of processes (passed by *-np* parameter of `mpirun`) should be set to a number of virtual cores plus one (e.g. use `-np 9` on a machine with 8 cores). Please note that computing lower-bound trajectory scales very well (CaverDock can utilize a hundred of cores in a typical scenario), whereas upper-bound trajectory scales up to the number of concurrently executed docking set by `--parallel_workers_smooth`.
- Try increasing the number of parallel workers instead of exhaustiveness. The default numbers of processes solving the same docking scenario in parallel is 4. If one wants to increase the exhaustiveness of docking, the values of `--parallel_workers_lb` and `--parallel_workers_smooth` parameters may be increased instead of the value of `--exhaustiveness`.

The tips described above improve the CaverDock speed in general. However, some issues result from the properties of analyzed biochemical systems. We summarize typical issues and suggest possible solutions in the listing below.

- Backtracking is executed very frequently (this can be observed in CaverDock log files). When an upper-bound trajectory is computed, CaverDock tries to keep its energy as close to the lower-bound as possible. In some cases, it is not possible and CaverDock executes a lot of backtracking trying to find a better trajectory without any success. The number of executed backtracking runs can be decreased by parameters `--backtrack_threshold` (higher difference of the lower-bound and upper-bound energies may be required to start backtracking) or `--backtrack_limit` (the frequency of backtracking execution may be lowered). Beware that usually high computational times and high number of backtracking executions are related to suboptimal geometry of the tunnel or the receptor (Section 6.2 for geometry optimization tips).

- The number of degrees of freedom is very high. If the number of flexible side-chain residues is high, we can fix some residues in such a position which does not form a bottleneck and make them rigid. We may also try to compute a contiguous ligand movement with a non-contiguous movement of side-chain residues by using parameter `--allow_flex_discontinuity` (which removes some constraints in the search space). If the origin of the high number of degrees of freedom is mainly the complexity and the size of a ligand, it is possible that this system is too complex to be efficiently analyzed by CaverDock.

7 Troubleshooting

7.1 CaverDock cannot compute the lower-bound trajectory

There are several possible reasons, which may prevent CaverDock from computing the lower-bound trajectory.

- A part of the tunnel is too narrow and therefore it forms a strong repulsive barrier. This situation needs manual inspection: CaverDock saves only the part of the lower-bound trajectory which was successfully computed. The missing part of the trajectory probably contains a strong repulsive barrier such as a residue preventing the ligand from moving through the tunnel. The receptor geometry needs to be fixed by adding flexibility to side-chain residues, or using a different snapshot of the receptor.

- The ligand is forced to move against the active site bottom. This problem arises from a geometrical analysis of the tunnel where the geometrical approximation of the tunnel is too deep. An easy solution is to use the `--dock_like` parameter to navigate CaverDock where to start with the tunnel analysis.

- The ligand is too complex to be successfully docked. CaverDock can be re-executed with higher exhaustiveness, or a higher number of parallel workers.

- The computation fails due to the stochastic nature of CaverDock. In this case, starting CaverDock once again should solve the problem.

7.2 CaverDock cannot compute the upper-bound trajectory

When an upper-bound trajectory cannot be computed, the user should inspect the lower-bound trajectory first. When the lower-bound trajectory already contains a high energetic

barrier, the tunnel or the receptor geometry needs to be modified (Section 6.2). If it is not possible to improve the lower-bound trajectory, the selected ligand is likely not able to pass through the tunnel.

When the lower-bound trajectory does not contain any significant barrier and the upper-bound trajectory is still not computed, CaverDock is either not able to analyze the trajectory because of a high ligand complexity, or there is a bottleneck not detected by the lower bound trajectory. This type of bottlenecks can be found by inspecting the lower-bound trajectory in the vicinity of the disc, where the computation of the upper-bound trajectory has failed[3]. There should be visible non-contiguities in the lower-bound trajectory, which overpass the bottleneck (e.g. ligand flip, as shown in Fifure 3). The bottleneck needs to be manually identified and fixed (e.g. by using flexible side-chain residues).

8 FAQ

8.1 Is CaverDock execution deterministic?

In the default settings, it is not deterministic (so re-executing CaverDock may result in different trajectories). There is, however, a way how to make CaverDock deterministic. It must be executed in two processes only (using `mpirun -np 2`) and the random number generator must be set to a constant seed (using `--seed x`, where `x` is any number, which must be the same for all deterministic executions). Please note than when CaverDock is executed only for single docking (possibly with constraints), setting a constant seed is sufficient.

8.2 Why does CaverDock return two types of energy?

The CaverDock computation of an upper-bound trajectory is driven by a heuristics. It means that CaverDock cannot guarantee that the contiguous trajectory is optimal. Thus, CaverDock is computing also a lower-bound trajectory: a scenario, which can be unrealistically optimistic due to non-continuities in the trajectory. The real energy (with respect to the given force-field and the geometry of the input molecules) is at most as high as the energy of the upper-bound trajectory and at least as low as the energy of the lower-bound trajectory.

References

[1] O. Trott, A. J. Olson, AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization and Multithreading. *Journal of Computational Chemistry 31 455-461 (2010).*

[2] O. Vavra, J. Filipovic, J. Plhak, D. Bednar, S. Marques, J. Brezovsky, L. Matyska, J. Damborsky, CAVERDOCK: A New Tool for Analysis of Ligand Binding and Unbinding Based on Molecular Docking. *In preparation.*

[3] J. Filipovic, O. Vavra, J. Plhak, D. Bednar, S. Marques, J. Brezovsky, L. Matyska, J. Damborsky, A Novel Method for Analysis of Ligand Binding and Unbinding Based on Molecular Docking. *In preparation.*

[3] CaverDock reports a number of the last disc, for which the upper-bound trajectory was computed. The number of snapshot in the lower-bound trajectory is equal to the number of the disc.

[4] E. Chovancova, A. Pavelka, P. Benes, O. Strnad, J. Brezovsky, B. Kozlikova, A. Gora, V. Sustr, M. Klvana, P. Medek, L. Biedermannova, J. Sochor, J. Damborsky, CAVER 3.0: A Tool for Analysis of Transport Pathways in Dynamic Protein Structures. *PLOS Computational Biology 8: e1002708 (2012).*

[5] CAVER Web Portal. https://loschmidt.chemi.muni.cz/caverweb/

[6] CAVER User Guide. http://www.caver.cz/fil/download/manual/caver_userguide.pdf

[7] MGLTools. http://mgltools.scripps.edu/

[8] AutoDock Vina Users Guide. http://vina.scripps.edu/manual.html