Chapter 3

Computational Analysis of Protein Tunnels and Channels

Jan Brezovsky, Barbora Kozlikova, and Jiri Damborsky

Abstract

Protein tunnels connecting the functional buried cavities with bulk solvent and protein channels, enabling the transport through biological membranes, represent the structural features that govern the exchange rates of ligands, ions, and water solvent. Tunnels and channels are present in a vast number of known proteins and provide control over their function. Modification of these structural features by protein engineering frequently provides proteins with improved properties. Here we present a detailed computational protocol employing the CAVER software that is applicable for: (1) the analysis of tunnels and channels in protein structures, and (2) the selection of hot-spot residues in tunnels or channels that can be mutagenized for improved activity, specificity, enantioselectivity, or stability.

Key words Binding, Protein, Tunnel, Channel, Gate, Rational design, Software, CAVER, Transport

1 Introduction

All biomolecules contain a complex system of voids—cavities, channels, tunnels, or grooves of various shapes and sizes (Fig. 1a). Cavities often host a site of functional importance, but in many cases, also tunnels and channels have functional roles. They secure the transport of ligands between different regions, e.g., connect buried cavities with the surface, different cavities, or even different cellular compartments, such as in membrane proteins. The presence of tunnels was already reported for numerous enzymes from all six Enzyme Commission classes as well as all main structural classes [1, 2]. The geometry, physicochemical properties, and dynamics of tunnels have been shown to determine the exchange rates of ligands between the active sites and a bulk solvent. Additional functions are often secured by the tunnels: (1) enabling the access of preferred substrates, while denying the access of nonpreferred ones and thus preventing the formation of nonproductive complexes, (2) avoiding the damage of the enzymes dependent on cofactors containing the transition metals by their poisoning, (3) preventing the damage to the cell by releasing toxic intermediates,

Uwe T. Bornscheuer and Matthias Höhne (eds.), *Protein Engineering: Methods and Protocols*, Methods in Molecular Biology, vol. 1685, DOI 10.1007/978-1-4939-7366-8_3,

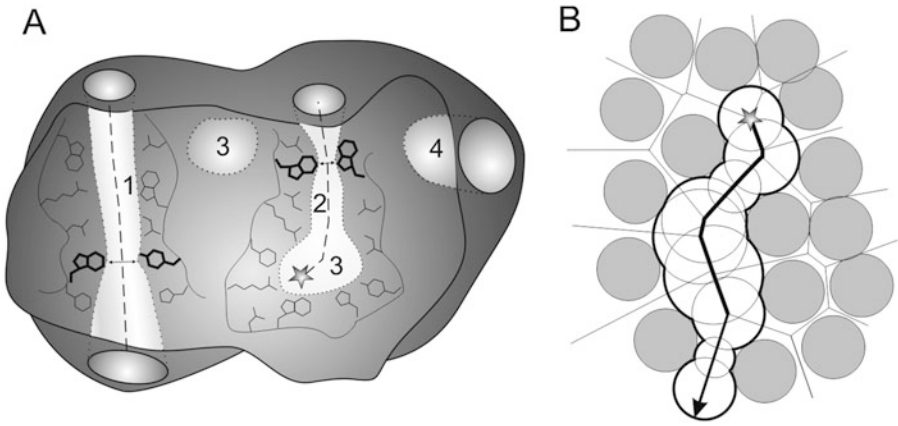

Fig. 1 (**a**) Schematic representation of complex voids in protein structures. Channel (1), tunnel (2), buried cavities (3), and surface groove (4) are in *light gray*. Channel and tunnel entrance is in *gray*. Channel and tunnel centerlines are shown as *dashed lines*, bottlenecks are indicated by *black arrows*, and bottleneck-lining residues are in *bold*. The *gray* star in the buried cavity represents the starting point for the tunnel calculation. Cavity-, channel-, and tunnel-lining residues are in *black* lines, while bottleneck residues are in *black* sticks. (**b**) Schematic representation of Voronoi diagram. Atoms are represented by *gray circles*, edges by *lines*, a starting point of tunnel by a *gray star*, a tunnel exit by an *arrow*, a tunnel surface by a *thick black contour*

(4) enabling reactions that require the absence of water, and (5) synchronizing reactions that require the contact of a large number of substrates, intermediates, or cofactors [1, 3]. Recognizing the importance of transport processes for enzymatic catalysis, the key–-lock–keyhole model has recently been proposed [1] and experimentally validated by a number of protein engineering studies. These studies successfully exploited the tunnel modification for improving enzyme activity [4, 5], specificity [6], enantioselectivity [7], and stability [8, 9]. Identification of tunnels and channels in the complex voids present in protein structures is not a trivial task, and many software tools have recently been developed for this purpose, e.g., CAVER [10], MOLE [11], MolAxis [12], ChExVis [13], or BetaCavityWeb [14]. All these tools are based on computational geometry methods employing the Voronoi diagrams. These methods identify the pathways connecting a starting point in a buried cavity to a bulk solvent in the target protein structure (Fig. 1b). The starting point is defined by several atoms or residues surrounding an empty space of the occluded protein cavity. As the main output, the tools provide geometry of the tunnels, which have their minimal radius wider than the radius specified by the user. The tunnels' geometry is supplemented by the information about their characteristics, profiles, the tunnel-lining residues, and the residues forming the tunnel bottleneck, i.e., the narrowest part of the access tunnel [15]. The bottleneck residues represent particularly suitable hot-spots for the modification of tunnels' geometry since their

substitutions often have substantial impact on the function or stability of the protein [1, 3].

The software tools outlined in the previous paragraph enable fast identification of permanent tunnels in a single structure and provide information about their characteristics. The main limitation of such analysis is that protein dynamics is neglected and the transient tunnels are missed. In other words, only the tunnels that are open in the analyzed structure are identified, while the transient gated tunnels, which are temporarily closed in the investigated structure, can be overlooked [10, 16]. Additionally, it is often difficult to distinguish biologically relevant tunnels using the analysis of a single static structure [10, 16]. Both pitfalls can be overcome by analyzing the tunnels in an ensemble of protein conformations. These conformations can be obtained from: (1) NMR ensemble, (2) set of crystal structures, or (3) molecular dynamics simulations (*see* **Note 1**). From the aforementioned tools, only CAVER was designed for comprehensive analysis of tunnels in molecular ensembles and includes essential tunnel clustering [15]. However, setting up molecular dynamics simulations requires considerable expertise and knowledge of a studied protein. Hence, the description of molecular dynamics and the analysis of tunnels over an ensemble is beyond the scope of this chapter (*see* **Note 2**).

2 Overview of Implementations of CAVER 3.0

CAVER is a software tool widely used for the identification and analysis of transport tunnels in macromolecular structures. It is available in several implementations that provide various sets of features (Table 1), addressing needs of users with different level of experience and expectations: (1) a command-line application for analysis of both static and dynamic structures, (2) a PyMOL plugin for the analysis of static structures, and (3) a graphical user interface CAVER Analyst [17] for the analysis of both static and dynamic structures as well as the visualization of detected tunnels. Additionally, CAVER is also integrated into two web services: (4) CAVER Web (under preparation), (5) HotSpot Wizard [18] and (6) CaverDock (under preparation). These interactive web tools are accessible via the site: http://loschmidt.chemi.muni.cz/peg/software.

1. **CAVER Command-line application** is suited for advanced users who aim to analyze large ensembles of structures, typically counting thousands or tens of thousands of snapshots. The larger number of structures may require the use of supercomputers. Moreover, the command-line application is employed as tunnel analysis engine in all other implementations. The setting of the tunnel calculation is performed via a

Table 1
Main features of various implementations of CAVER 3.0

Feature	CAVER command-line	CAVER PyMOL plugin	CAVER Analyst	CAVER Web	HotSpot Wizard
Input format	PDB	Formats supported by PyMOL	PDB, mdcrd, xtc, and dcd	PDB	PDB
Analysis of ensembles	+	–	+	–	–
Assistance during preparation of input structure	–	–	+	+	+
Manipulation of protein structure	–	+	+	–	–
Additional analyses of protein structure	–	+	+	–	–
Advanced settings of calculation	+	+	+	–	–
Start from–user-defined point	+	+	+	+	–
Start from–buried cavity	–	–	+	+	+
Start from–catalytic residues	–	–	+	+	+
Direct visualization of tunnels	–	+	+	+	+
Interactive GUI	–	–	+	+	–
Interactive exploration of output statistics	–	–	+	+	–
Advanced visualization	–	–	+	–	–
Detailed information about tunnels properties	+	+	+	+	–
Physicochemical properties of tunnels	–	–	+	+	–
Evolutionary conservation	–	–	–	+	+

configuration file, and the visualization of the tunnels is enabled by scripts for PyMOL and VMD programs, or by using CAVER Analyst. Detailed information about the tunnels, the residues, and the atoms surrounding these tunnels, and the bottleneck residues are provided in the text files. The software is freely available at www.caver.cz.

2. **CAVER PyMOL plugin** is suited for unexperienced users who need a simple exploration of tunnels or channels in their

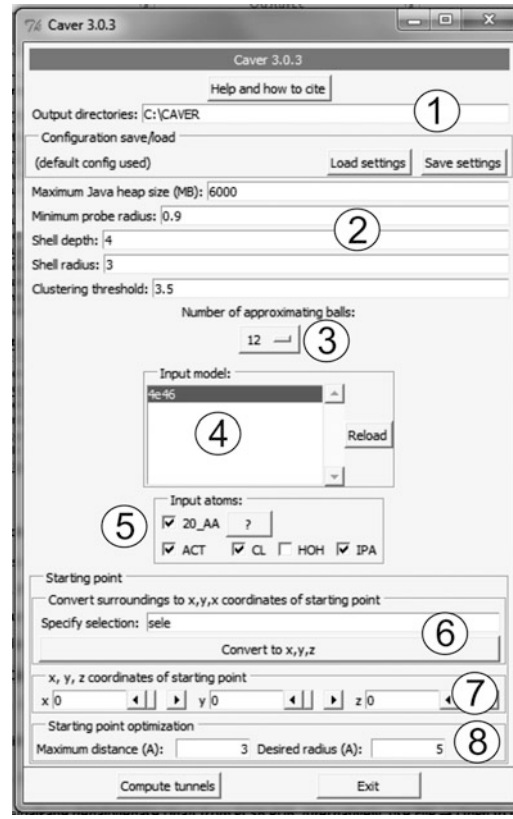

Fig. 2 Input form of CAVER plugin for PyMOL. Individual sections described in the protocol are numbered. 1–output directories, 2–tunnel calculation parameters, 3–number of approximating balls, 4–list of input model for analysis, 5–input atoms, 6–selection defining a starting point, 7–coordinates of the starting point, 8–parameters for the starting point optimization

biomolecular structures from crystallographic or NMR analyses. The plugin provides a graphical interface for setting up the calculation and interactive visualization of results (Figs. 2 and 3). The starting point position can be specified by using any PyMOL selection or alternatively by its coordinates. The calculation can be performed for any structure loaded into PyMOL. Similarly to the command-line application, additional information is available from the text files. The software is freely available at www.caver.cz.

3. **CAVER Analyst** provides the users with the interactive multiplatform environment for a comprehensive analysis on both static and dynamic structures (Fig. 4a). It allows interactive exploration of tunnels computed either by the command-line version or directly from the CAVER Analyst interface. It contains additional features for evaluation of the biological relevance of tunnels (calculation of physicochemical properties,

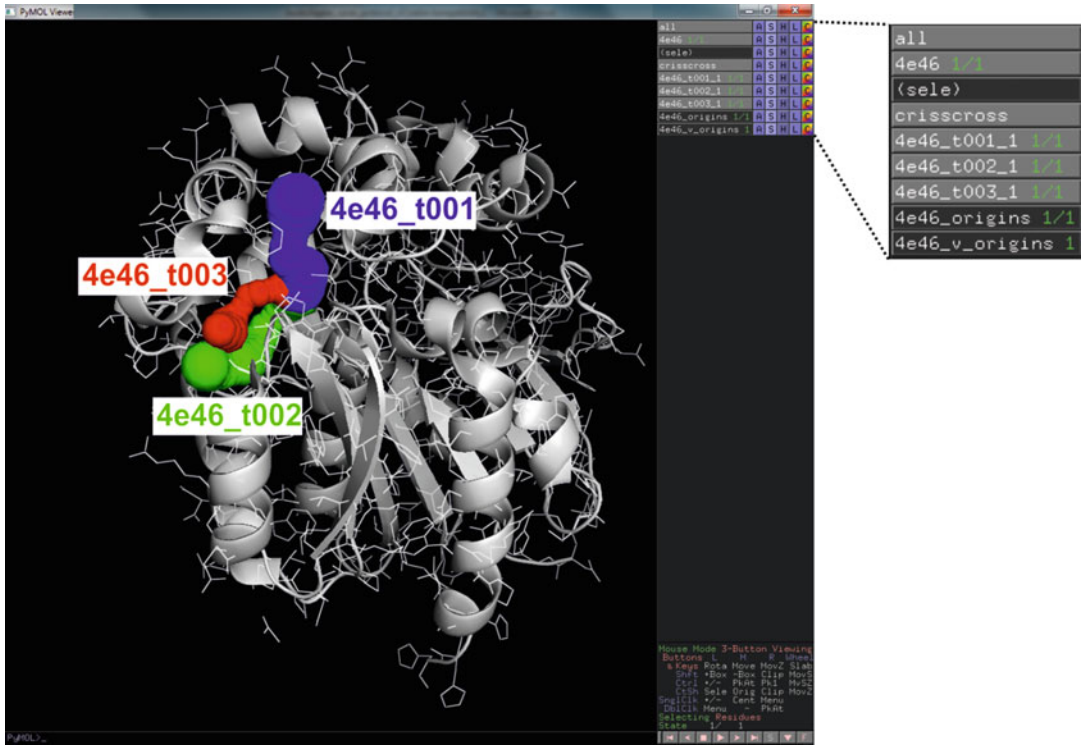

Fig. 3 Visualization of the tunnels in the haloalkane dehalogenase DhaA by the CAVER plugin for PyMOL. The protein is shown as *gray* cartoon and individual tunnels as colored spheres. The tunnels are available as separate PyMOL objects labeled as *4e46_tX*, where X represents the tunnel ID, and are ordered accordingly to their throughputs

filtering of tunnels by their parameters, etc.) and for comparative analysis of tunnels in homologous structures. The setup of the calculation is facilitated by cavity calculations and querying databases for functionally relevant residues. It accepts ensembles in Protein data bank (PDB) format, as well as formats of several molecular dynamics programs: *mdcrd* of AMBER [19], *xtc* of GROMACS [20], and *dcd* of CHARMM [21]. The software is freely available at www.caver.cz.

4. **CAVER Web** is a web server providing intuitive, straightforward, and fast analysis of tunnels for inexperienced users (Fig. 4b). The user is guided through the preparation of the input structure and setting of the tunnel calculations in a step-by-step manner, maximizing the biological relevance of obtained results with minimal effort. The visualization of tunnel geometry using JSMOL is provided along with several basic tables and graphs summarizing the most important properties of the identified tunnels. The full version is freely available at: http://loschmidt.chemi.muni.cz/caverweb.

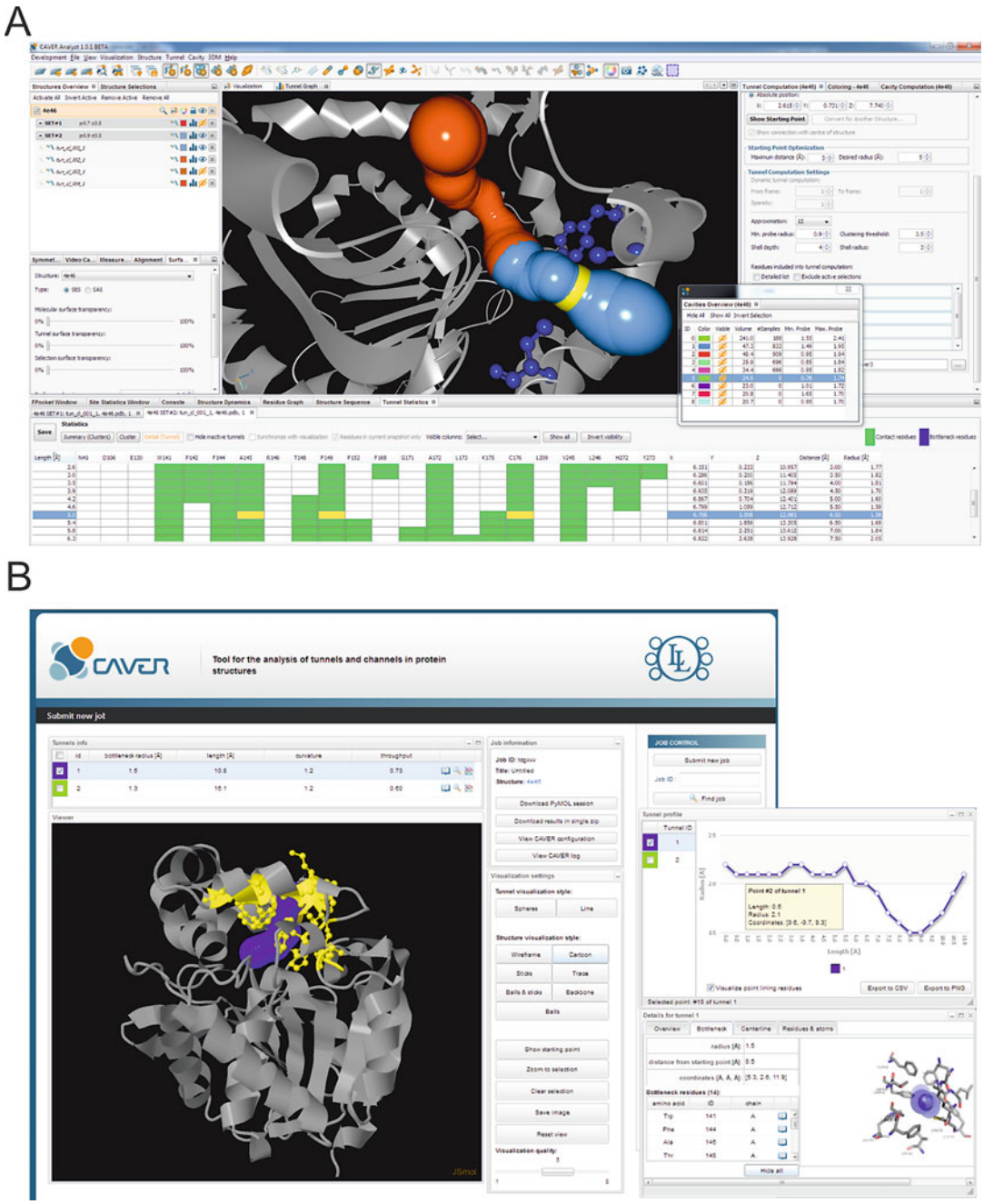

Fig. 4 Graphical user interfaces of CAVER for interactive analysis of tunnels and channels. (**a**) CAVER Analyst and (**b**) CAVER Web

5. **HotSpot Wizard** is a web server focused on automated prediction of hot-spot residues for mutagenesis that are likely to alter enzyme activity, selectivity, or stability. Within its workflow, tunnels are calculated from the automatically identified

active site cavity. Tunnels are visualized with their basic parameters, e.g., bottleneck radius and length, and the identity of tunnel-forming residues is provided. Additionally, the information on the evolutionary conservation of the tunnel-forming residues is provided. The server is freely available at: http://loschmidt.chemi.muni.cz/hotspotwizard.

6. **CAVER Dock** is a software tool for rapid analysis of transport processes in proteins. It models the transportation of a ligand from outside environment into the protein active or binding site and vice versa. The input is a protein structure in PDB format and a ligand structure in the PDBQ format. The outputs are ligand's trajectory and energetic profile. CAVER Dock implements a novel algorithm which is based on molecular docking and is able to produce contiguous ligand trajectory and estimation of a binding energy along the pathway. It uses CAVER for pathway identification and heavily modified AUTODOCK VINA as a docking engine. The tool is much faster than molecular dynamic simulations (2-20 min per job). The software is easy to use as it requires in its minimalistic configuration the setup for AUTODOCK VINA and geometry of the tunnel. The tool is freely available at: http://loschmidt.chemi.muni.cz/caverdock.

3 Protocol for Analysis of Tunnels by CAVER

In this example, we will analyze tunnels in a crystal structure of haloalkane dehalogenase DhaA (PDB-ID 4e46) using the CAVER plugin for PyMOL. The aim of this analysis is to: (1) find all tunnels present in the structure, (2) select the most functionally relevant tunnels, and (3) identify residues forming these tunnels and their bottlenecks as the most promising targets for protein engineering.

3.1 Calculation Setup

1. In the PDB database, select a protein structure, ideally with high resolution, without missing atoms or residues and with a biologically relevant quaternary structure (*see* **Note 3**). The structure of the haloalkane dehalogenase DhaA with PDB-ID 4e46 determined to the resolution 1.26 Å by protein crystallography is a suitable choice.
2. In PyMOL, use *Plugin* → *PDB Loader Service* → 4e46 to download the PDB file of the haloalkane dehalogenase DhaA from RCSB PDB. Alternatively, use *File* → *Open* to load the PDB file from the local computer (*see* **Note 4**).

3. Use *Plugin* → *Caver 3.0.x* to invoke the CAVER plugin for calculation of tunnels (Fig. 2). In case the plugin is not available, *see* **Note 5** for further instructions.
4. In the *Output directories* field specify the path to the directory where the results of the CAVER analysis should be stored.
5. Set the parameters for calculation of tunnels:
   (a) *Maximum Java heap size*—specifies the maximum memory allocated for computation (*see* **Note 6**). The default value of 6000 MB is more than enough to enable the analysis of the *4e46* structure, however this structure can be processed even with as little as 500 MB of the memory.
   (b) *Minimum probe radius*—defines the minimum radius of the identified tunnels. For permanently opened tunnels, this parameter should approximately correspond to the dimension of the transported ligands. However, smaller radii may be more appropriate for transient/gated tunnels to emulate missing protein dynamics. For discussion on drawbacks of using smaller probe, *see* **Note 7**. In our case, the default value of 0.9 Å will be sufficient to disclose all main ligand pathways.
   (c) *Shell radius* and *Shell depth* parameters—define the molecular surface of the protein and ultimately the endpoints of the identified tunnels. The protein surface is delineated by rolling a probe of the *Shell radius* over the protein structure. Smaller S*hell radius* provides less approximate description of the protein surface, but it could also disallow the identification of the appropriate tunnels (*see* **Note 8**). The *Shell depth* parameter disables the branching of tunnels within a given depth from the protein surface, defined by the *Shell radius*. Suitable values for these two parameters significantly differ between individual proteins and need to be optimized for each case (*see* **Note 8**). The default values (*Shell depth* 4 Å and *Shell radius* 3 Å) are suitable for our example.
   (d) *Clustering threshold*—defines the level of details for tunnel branches. The smaller the *Clustering threshold*, the more branches will be provided. A very small value can result in the identification of many nearly identical tunnel branches. The default value of 3.5 is suitable for our example.
   (e) *Number of approximating balls*—specifies the number of balls that are employed to represent individual atoms in the input structure of protein to enable the construction of an ordinary Voronoi diagram. Using a high number of balls increases the accuracy of results, but also the

computational time and memory demands. For most of the cases, the usage of default 12 balls provides acceptable precision.

6. Select *4e46* as the *Input model* out of a list of input models available to perform the tunnel analysis of this structure.
7. In the *Input atoms* section, choose only the relevant atoms belonging to the macromolecule to be employed in the analysis (*see* **Note 9**). In this example, use 20 standard amino acid residues (20_AA) as the only atoms included in the calculation. Remaining ligands named ACT, CL, and IPA have to be deselected. Water molecules (HOH) are excluded automatically.
8. Specify the starting point for the tunnel calculation. The starting point is initially placed in the center of mass of the residues or atoms specified by PyMOL selection. Residues suitable for the creation of such a selection are: (1) residues forming the relevant cavity, (2) catalytic residues in the cavity, or (3) ligands in the cavity (*see* **Note 10**). It is important to stress that an appropriate starting point has to be located in an empty space of the cavity and cannot intersect with any protein atoms (*see* **Note 11**). In our example, select the ligand with residue name IPA, which is conveniently located at the center of the active site (*see* **Note 12**). Once a suitable selection is created in PyMOL, the user has to input the name of this selection into the S*pecify selection* field and press the *Convert to x, y, z* button. This will show the initial location of the starting point as a white cross. At this stage, the user has to check if the point is located inside the empty space within the protein structure. If it is not, the user can either manually modify the position of the starting point by changing the absolute coordinates, or rely on the automatic optimization of the position that is controlled by the *Maximum distance* and *Desired radius* parameters. This optimization will relocate the starting point to the empty sphere of the *Desired radius* up to the *Maximum distance* from its initial position (*see* **Note 13**).
9. Press *Compute tunnels* button to start calculation of tunnels in the structure of the haloalkane dehalogenase DhaA.

3.2 Interpretation of Results

1. Explore the tunnels identified in the context of the haloalkane dehalogenase DhaA structure (Fig. 3). The identified tunnels become available as separate PyMOL objects labeled as *4e46_tX*, where X represents tunnel IDs, and ordered accordingly to the tunnel *throughput*, i.e., a metrics combining the length and width of the tunnel. The throughput reflects the predicted ability of a tunnel to transport small molecules. Individual tunnels can be shown/hidden and their visualization style modified by using the right control panel of PyMOL.

Table 2
Main geometric parameters of the tunnels identified in the haloalkane dehalogenase DhaA

ID	Avg_BR [Å]	Avg_L [Å]	Avg_C	Avg_throughput
1	1.57	9.18	1.13	0.76
2	1.28	16.11	1.49	0.62
3	1.06	10.69	1.25	0.59

Avg_BR average bottleneck radius, *Avg_L* average tunnel length, *Avg_C* average tunnel curvature, *Avg_throughput* average tunnel throughput. Please note that in the case of static protein structure, these values are averaged over a single tunnel, i.e., these parameters correspond directly to the bottleneck radius, tunnel length, and curvature of a particular tunnel

2. Extract the main geometric parameters of the identified tunnels from the *summary.txt* file, which is located in the *Output directories* (*see* **Note 14**). The most important parameters of each tunnel are the bottleneck radius (*Avg_BR*), the tunnel length (*Avg_L*), the tunnel curvature (*Avg_C*), and the tunnel throughput (*Avg_throughput*) (Table 2).
3. Combining the visualization of the tunnels with the information about their parameters, the relevance of each tunnel with respect to protein function can be estimated (*see* **Note 15**). In our case, the first tunnel (*4e46_t001*), with the bottleneck radius of nearly 1.6 Å, is the only tunnel wide enough to enable the transport of any molecule without a need for conformational changes in the protein structure. Therefore, this tunnel will be in focus of our further analyses. Note that two auxiliary tunnels could still represent the suitable targets for engineering—especially of protein stability [8] and activity [5].
4. Perform an analysis of the tunnel profile by plotting the tunnel radius against the tunnel length. This enables identification of tunnel bottlenecks. For this purpose, use the data from the *caver_output/calculation_id/analysis/tunnel_profiles.csv* file, located in the previously defined *Output directories* (*see* **Note 14**). Focusing on the tunnel cluster 1, plot *R* versus *length* parameters (Fig. 5a). One can easily identify the location of the tunnel bottleneck 5.5 Å from the starting point from this graph (*see* **Note 16**).
5. Identify the tunnel- and the bottleneck-forming residues representing the potential hot-spots for mutagenesis (*see* **Note 17**). Open the *caver_output/calculation_id/analysis/residues.txt* file from the *Output directories* (*see* **Note 14**). There are 22 residues lining the tunnel 1 (Fig. 5b). Out of these, 18 residues contribute to the formation of the tunnel by their side chains, while the remaining four residues participate only by backbone with their side chains oriented outward the tunnel. The former residues represent good hot-spots for modulation of the

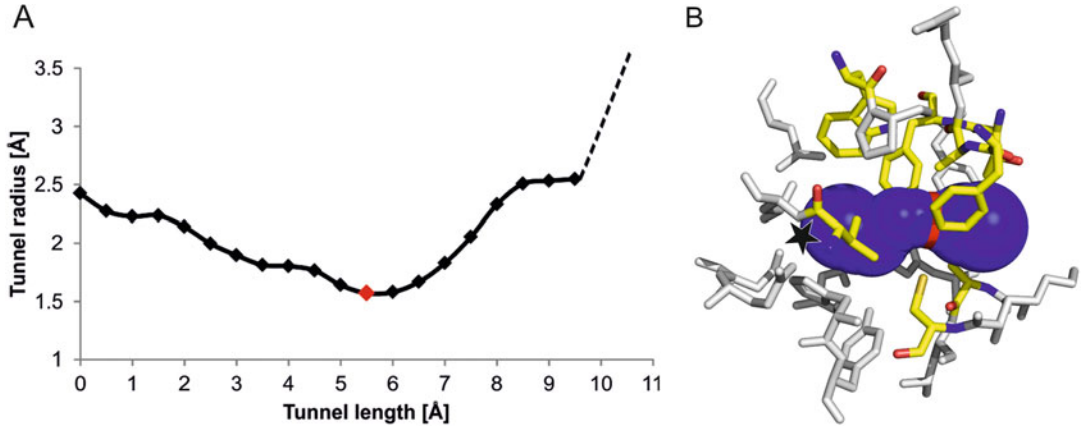

Fig. 5 Analysis of the main tunnel of the haloalkane dehalogenase DhaA. (**a**) Profile of the main tunnel. The position of the bottleneck is represented by the *red* diamond in the profile. The dashed-line illustrates the further widening of the pathway at the protein surface. (**b**) Tunnel-lining and bottleneck residues identified by CAVER. Residues are shown as *white* (tunnel-lining) and *yellow* (bottleneck) sticks

properties of the tunnel by substitutions. However, one has to be careful since some of these residues form also the wall of the buried cavity, and thus are often involved in other protein functions, such as ligand binding and catalysis. These indispensable residues must be excluded from further considerations to provide a viable design (*see* **Note 18**). A safer alternative is to focus on the bottleneck residues that are less frequently overlapping with the catalytic or ligand binding residues. At the same time, their modification has higher probability to alter the tunnel properties [10]. The list of residues forming the tunnel bottleneck is available in the file *caver_output/calculation_id/analysis/bottlenecks.csv*, located in the *Output directories* (*see* **Note 14**). In our case, there are eight bottleneck residues (Fig. 5b): Trp141, Phe144, Ala145, Thr148, Phe149, Ala172, Cys176, and Val245. Referring to the *residues.txt* file, the user can verify that all these residues contribute to the tunnel by their side chain, and thus could be considered as proper hot-spots.

4 Notes

1. Individual structures of the ensemble must be aligned to a selected reference structure, located in a single directory, and in the case of time-dependent ensemble also consecutively numbered in a computer-recognized order (e.g., the structure "10" could come before "2" thus it is necessary to number the

structures as "02" and "10" instead). For NMR structures available from the PDB database, the individual models present in the PDB file must be split into separate files. The detailed protocol for the analysis of ensemble data is described in the User guides of CAVER 3.0 and CAVER Analyst 1.0.

2. The molecular dynamics trajectories of some proteins could be procured from databases like MoDEL—http://mmb.pcb.ub.es/MoDEL [22].
3. The PDB file of the analyzed protein has to be of sound quality, hence special attention should be devoted to its resolution, information on missing atoms or residues, and its biological unit. Structures with resolution below 2 Å are preferred for the analysis. However, beware of the alternative conformation of residues that are frequently present in high-resolution structures, as CAVER retains only those conformations with the highest occupancy by default. Missing atoms or residues could lead to the identification of artificial tunnels. Using an asymmetric unit instead of a proper biological unit of the protein could also be a source of errors: (1) artificial tunnels are identified that lead through a space that is occupied by another protein chain of the biological unit, and (2) some relevant tunnels may not be identified because their openings at the surface can be blocked by other protein chain, that would not be present in the biological unit.
4. Aside from the PDB format supported by the CAVER plugin and the CAVER command-line application, additional input formats from molecular dynamics trajectories are supported in CAVER Analyst (AMBER—*mdcrd*, GROMACS—*xtc*, and CHARMM—*dcd*).
5. The caver_3.0_plugin.zip file containing CAVER plugin for PyMOL and the user guide with instructions for the installation procedure can be obtained from the download section at http://www.caver.cz.
6. *Maximum Java heap size* parameter should be increased in the case when the *out of memory* error is encountered. When only a limited memory is available, the requirements can be reduced by decreasing the *number of approximating balls* parameter.
7. Using too small probe radius results in the identification of many artificial tunnels since narrow tunnels can be identified nearly everywhere in the protein structure. Due to a large number of such tunnels, the time required for their analysis increases notably. It is therefore preferred to analyze transient tunnels from molecular dynamics simulation using a probe of more realistic radius.
8. *Shell depth* and *Shell radius* parameters need to be optimized to avoid artificial tunnels (Fig. 6a). While the smaller S*hell radius*

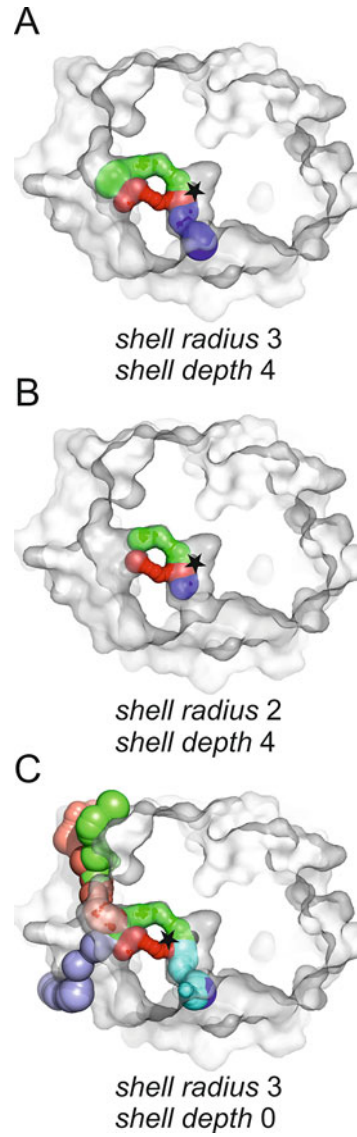

Fig. 6 Effect of different settings of the shell radius and shell depth parameters on tunnels of the haloalkane dehalogenase DhaA. (**a**) Optimal tunnels obtained by using the most suitable parameters. (**b**) Short tunnels poorly describing the real length due to too small *Shell radius*. (**c**) Artificial tunnels slithering on the protein surface due to too small *Shell depth.* The protein is shown as *grey surface*, individual tunnels as *colored spheres*, and the starting point is depicted as a *black star*

provides more realistic surface, it should also be larger than the maximum radius of the individual tunnels and must be larger than the bottleneck radius of each tunnel. Otherwise no tunnels or artificially shortened tunnels will be identified, since in such a case the protein interior or its part will be considered as bulk solvent (Fig. 6b). The user should increase the *Shell radius*

whenever the obtained tunnels are ending prematurely before reaching the protein surface. A hallmark of too large *Shell radius* and too small *Shell depth* is the presence of many tunnels that seem to be slithering on the protein surface (Fig. 6c). In case the *Shell radius* cannot be further decreased and many slithering tunnels are still identified, the user should increase the *Shell depth* instead. Since the *Shell depth* prohibits branching of tunnels, it is necessary to check that all important tunnel branches are identified under such settings. Conversely, if some important tunnel or some important tunnel branch is missing, the *Shell depth* should be decreased.

9. In general, only atoms of the investigated biomolecules, i.e., amino acid residues or nucleotides, should be included in the analysis, as the ligand moieties present in the structures could occlude the tunnels and disallow their identification. However, special care has to be taken with covalently bound atoms of modified residues, cofactors, or metals that should be included in the analysis as integral components of the biomolecule.
10. Automatic identification of catalytic residues and occluded cavities can be performed by CAVER Analyst, CAVER Web, or Hotspot Wizard. Alternatively, the user can perform a manual analysis of cavities and important residues. There are many tools serving for the identification and analysis of pockets in protein structure [23, 24]. Some of these tools, e.g., CASTp [25], MetaPocket 2.0 [26], or Fpocket [27], are available as easy-to-use web servers, providing a list of residues forming each identified pocket. To recognize the relevant pockets, the user can employ the information on catalytic residues in CASTp, a consensus prediction of several tools in MetaPocket 2.0, or a druggability score provided by Fpocket. If no relevant pocket can be unambiguously selected, the information on functional or catalytic residues from UniProtKB [28] or Catalytic Site Atlas 2.0 [29] databases can be employed for specifying the functional pocket in which these residues are localized.
11. When using the protein residues to define the starting point, more than one residue from the protein is often required for the optimal placement of the point into an empty space. By clicking on three residues around a cavity, PyMOL will create the selection named "sele". Atoms included in this selection will be used by CAVER to calculate a starting point as their geometric center upon pressing the *Convert to x, y, z* button. The readers may want to try this approach, e.g., by selecting Asp106, Leu209 and Phe168 residues.
12. It is important to note that defining the starting point by the ligand IPA will work only in the case that the atoms of the

ligand were excluded from the analysis during the **step 7**, as exemplified by our protocol.

13. Increasing the *Desired radius* parameter will place the starting point in the center of the closest empty space large enough to harbor a sphere of at least *Desired radius*, within the *Maximum distance*. The potential pitfall of setting both *Maximum distance* and *Desired radius* parameters to relatively high values is that such settings could displace the calculation starting point outside the protein structure or to a different cavity, and thus no tunnels will be identified. After the completion of the tunnel calculation, the position of the optimized starting point will be available in the output folder (*see* **Note 14**) as the PyMOL object *4e46_v_origins*.
14. Individual output files with details about the CAVER analysis are located in the *Output directories* specified during the fourth step of the calculation setup, in the folder *caver_output/calculation_id*. *Calculation_id* subfolder with the highest number corresponds to the latest analysis.
15. The properties that may account for higher functional relevance of the tunnels are larger width, shorter length, and less curved trajectory [10]. The relevant tunnels are often composed of residues with physicochemical properties complementary to their cognate ligands [30]. The bottleneck residues are often flexible to enable a tunnel gating [16].
16. In some cases, the tunnel bottleneck may be detected at the beginning of the tunnel, which indicates that the starting point was incorrectly placed too close to the protein atoms. Such problem can be solved by moving the starting point into an empty space either manually or by adjusting the *Maximum distance* and the *Desired radius* parameters.
17. Both the tunnel-lining and the bottleneck-forming residues are identified by a predefined distance from the tunnel surface (3 Å by default). This approach may also provide false positive results, i.e., residues in the second shell of the tunnel or bottleneck. Therefore, without filtering over an ensemble of protein structures, the location and orientation of the identified residues should be visualized to verify their role in the tunnel or bottleneck formation.
18. The residues forming the walls of ligand-binding or catalytic pockets, as well as catalytic residues, can be identified by the tools discussed (*see* **Note 10**). The residues in a contact with bound ligands present in the structure could be identified by LigPlot+ [31], PoseView [32], or LPC [33]. Alternatively, users can employ an integrative analysis by HotSpot Wizard [18]. A detailed discussion on the construction of smart libraries can be found in the book chapter by Sebestova et al. in the recent Methods in Molecular Biology series [34].

Acknowledgments

The authors would like to express their thanks to Sergio Marques and David Bednar (Masaryk University, Brno) and to the editors Uwe Bornscheuer and Matthias Höhne (University Greifswald, Greifswald) for critical reading of the manuscript. MetaCentrum and CERIT-SC are acknowledged for providing access to supercomputing facilities (LM2015042 and LM2015085). The Czech Ministry of Education is acknowledged for funding (LQ1605, LO1214, LM2015051, LM2015047 and LM2015055). Funding has been also received from the European Union Horizon 2020 research and innovation program under the grant agreement No. 676559.

References

1. Prokop Z, Gora A, Brezovsky J et al (2012) Engineering of protein tunnels: keyhole-lock-key model for catalysis by the enzymes with buried active sites. In: Lutz S, Bornscheuer UT (eds) Protein engineering handbook. Wiley-VCH, Weinheim, pp 421–464
2. Gora A, Brezovsky J, Damborsky J (2013) Gates of enzymes. Chem Rev 113:5871–5923
3. Kingsley LJ, Lill MA (2015) Substrate tunnels in enzymes: structure-function relationships and computational methodology. Proteins 83:599–611
4. Biedermannova L, Prokop Z, Gora A et al (2012) A single mutation in a tunnel to the active site changes the mechanism and kinetics of product release in haloalkane dehalogenase LinB. J Biol Chem 287:29062–29074
5. Pavlova M, Klvana M, Prokop Z et al (2009) Redesigning dehalogenase access tunnels as a strategy for degrading an anthropogenic substrate. Nat Chem Biol 5:727–733
6. Chaloupkova R, Sykorova J, Prokop Z et al (2003) Modification of activity and specificity of haloalkane dehalogenase from *Sphingomonas paucimobilis* UT26 by engineering of its entrance tunnel. J Biol Chem 278:52622–52628
7. Prokop Z, Sato Y, Brezovsky J et al (2010) Enantioselectivity of haloalkane dehalogenases and its modulation by surface loop engineering. Angew Chem Int Ed 49:6111–6115
8. Koudelakova T, Chaloupkova R, Brezovsky J et al (2013) Engineering enzyme stability and resistance to an organic cosolvent by modification of residues in the access tunnel. Angew Chem Int Ed 52:1959–1963
9. Liskova V, Bednar D, Prudnikova T et al (2015) Balancing the stability–activity tradeoff by fine-tuning dehalogenase access tunnels. ChemCatChem 7:648–659
10. Chovancova E, Pavelka A, Benes P et al (2012) CAVER 3.0: a tool for the analysis of transport pathways in dynamic protein structures. PLoS Comput Biol 8:e1002708
11. Sehnal D, Svobodova Varekova R, Berka K et al (2013) MOLE 2.0: advanced approach for analysis of biomacromolecular channels. J Cheminform 5:39
12. Yaffe E, Fishelovitch D, Wolfson HJ et al (2008) MolAxis: efficient and accurate identification of channels in macromolecules. Proteins 73:72–86
13. Masood TB, Sandhya S, Chandra N et al (2015) CHEXVIS: a tool for molecular channel extraction and visualization. BMC Bioinformatics 16:119
14. Kim J-K, Cho Y, Lee M et al (2015) BetaCavityWeb: a webserver for molecular voids and channels. Nucleic Acids Res 43:W413–W418
15. Brezovsky J, Chovancova E, Gora A et al (2013) Software tools for identification, visualization and analysis of protein tunnels and channels. Biotechnol Adv 31:38–49
16. Kingsley LJ, Lill MA (2014) Ensemble generation and the influence of protein flexibility on geometric tunnel prediction in cytochrome P450 enzymes. PLoS One 9:e99408
17. Kozlikova B, Sebestova E, Sustr V et al (2014) CAVER analyst 1.0: graphic tool for interactive visualization and analysis of tunnels and channels in protein structures. Bioinformatics 30:2684–2685

18. Pavelka A, Chovancova E, Damborsky J (2009) HotSpot wizard: a web server for identification of hot spots in protein engineering. Nucleic Acids Res 37:W376–W383
19. Case DA, Cheatham TE, Darden T et al (2005) The Amber biomolecular simulation programs. J Comput Chem 26:1668–1688
20. Berendsen HJC, van der Spoel D, van Drunen R (1995) GROMACS: a message-passing parallel molecular dynamics implementation. Comput Phys Commun 91:43–56
21. Brooks BR, Bruccoleri RE, Olafson BD et al (1983) CHARMM: a program for macromolecular energy, minimization, and dynamics calculations. J Comput Chem 4:187–217
22. Meyer T, D'Abramo M, Hospital A et al (2010) MoDEL (molecular dynamics extended library): a database of atomistic molecular dynamics trajectories. Structure 18:1399–1409
23. Henrich S, Salo-Ahen OMH, Huang B et al (2010) Computational approaches to identifying and characterizing protein binding sites for ligand design. J Mol Recognit 23:209–219
24. Perot S, Sperandio O, Miteva MA et al (2010) Druggable pockets and binding site centric chemical space: a paradigm shift in drug discovery. Drug Discov Today 15:656–667
25. Dundas J, Ouyang Z, Tseng J et al (2006) CASTp: computed atlas of surface topography of proteins with structural and topographical mapping of functionally annotated residues. Nucleic Acids Res 34:W116–W118
26. Zhang Z, Li Y, Lin B et al (2011) Identification of cavities on protein surface using multiple computational approaches for drug binding site prediction. Bioinformatics 27:2083–2088
27. Schmidtke P, Le Guilloux V, Maupetit J et al (2010) Fpocket: online tools for protein ensemble pocket detection and tracking. Nucleic Acids Res 38:W582–W589
28. UniProt Consortium (2015) UniProt: a hub for protein information. Nucleic Acids Res 43: D204–D212
29. Furnham N, Holliday GL, de Beer TAP et al (2014) The catalytic site atlas 2.0: cataloging catalytic sites and residues identified in enzymes. Nucleic Acids Res 42:D485–D489
30. Pravda L, Berka K, Svobodova Varekova R et al (2014) Anatomy of enzyme channels. BMC Bioinformatics 15:379
31. Laskowski RA, Swindells MB (2011) LigPlot+: multiple ligand-protein interaction diagrams for drug discovery. J Chem Inf Model 51:2778–2786
32. Stierand K, Rarey M (2010) Drawing the PDB: protein-ligand complexes in two dimensions. ACS Med Chem Lett 1:540–545
33. Sobolev V, Sorokine A, Prilusky J et al (1999) Automated analysis of interatomic contacts in proteins. Bioinformatics 15:327–332
34. Sebestova E, Bendl J, Brezovsky J et al (2014) Computational tools for designing smart libraries. Methods Mol Biol 1179:291–314