CAVER: Algorithms for Analyzing Dynamics of Tunnels in Macromolecules

Antonin Pavelka, Eva Sebestova, Barbora Kozlikova, Jan Brezovsky, Jiri Sochor and Jiri Damborsky

Abstract—The biological function of a macromolecule often requires that a small molecule or ion is transported through its structure. The transport pathway often leads through void spaces in the structure. The properties of transport pathways change significantly in time; therefore, the analysis of a trajectory from molecular dynamics rather than of a single static structure is needed for understanding the function of pathways. The identification and analysis of transport pathways are challenging because of the high complexity and diversity of macromolecular shapes, the thermal motion of their atoms, and the large amount of conformations needed to properly describe conformational space of protein structure. In this paper, we describe the principles of the CAVER 3.0 algorithms for the identification and analysis of properties of transport pathways both in static and dynamic structures. Moreover, we introduce the improved clustering solution for finding tunnels in macromolecules, which is included in the latest CAVER 3.02 version. Voronoi diagrams are used to identify potential pathways in each snapshot of a molecular dynamics trajectory and clustering is then used to find the correspondence between tunnels from different snapshots. Furthermore, the geometrical properties of pathways and their evolution in time are computed and visualized.

Index Terms—tunnel, pore, channel, pathway, macromolecule, molecular dynamics, CAVER, Voronoi diagram, Delaunay triangulation, average link hierarchical clustering

1 INTRODUCTION

Biological macromolecules, such as proteins and nucleic acids, play essential roles in life processes. An understanding of their structure and function is crucial for uncovering the principles of life [1], the development of new drugs [2] and applications in industry [3]. Computational analysis and visualization of macromolecules is important due to their complex structure, and high cost and limitations of laboratory analyses.

The function of a macromolecule often requires that a small molecule or ion is transported through its structure. The transport pathway usually leads through void spaces in a structure [4]. Properties of transport pathways change significantly in time, therefore the analysis of an ensemble of structures (e.g., trajectory of a molecular dynamics) rather than of a single static structure is needed for the understanding of tunnel function [5–9].

Previously, we released the CAVER 3.0 application and in [9] we discussed its biochemical relevance and briefly described the principles of calculations. This paper aims to provide the readers with the detailed description of the CAVER 3.0 algorithms that was not published before to offer the possibility for reproduction, comparison, and further development of these approaches. Furthermore, we present an improved clustering solution, which is available in the version CAVER 3.02 and allows to analyze very large sets of conformations containing hundreds of thousands of tunnels.

The paper first focuses on the input structures and the basic properties of their transport pathways and briefly introduces the Voronoi diagrams. Then the related work is discussed. The core of the paper is formed by the detailed description of all steps of our CAVER approach. Next, the properties of the detected tunnels and their visualization are discussed. Finally, the limitations of our approach are discussed and the future work is outlined.

1.1 Structures of Biological Macromolecules

The majority of the experimentally determined three-dimensional structures of proteins is archived in a single repository – the Protein Data Bank (PDB) [10]. The coordinates of individual atoms can be retrieved for most of the structures stored in the PDB. Structures containing thousands of atoms are the most
common, but some entries can contain even hundreds of thousands of atoms. The growth of the number of structures in the PDB is continuously increasing. At the end of 2014, the PDB held 105,400 structures and 9,651 new structures were deposited during that year, while 9,381 structures were deposited in 2013. A single conformation of a molecular structure is frequently modeled as the union of balls in \( \mathbb{R}^3 \). The dynamical behavior of a molecular system can be represented as time series of such unions of balls. Typically, atoms remain the same, and only their coordinates change in time. The ensembles of structures composed of hundreds of thousands of snapshots are not unusual.

### 1.2 Transport Pathways in Macromolecular Structures

Macromolecules can contain different types of inner pathways. The terms tunnel, channel, or pore denote the pathways in macromolecular structures that are used for the transport of small molecules and ions inside or through the structure. The terms channel and pore are most often used to describe transport pathways through biological membranes [9], [11]. The term tunnel can have two meanings. It denotes a pathway connecting two buried sites in a macromolecular structure [12] (e.g., different active sites in multi-functional enzymes). In the second case the tunnel is a pathway connecting a buried site (e.g., the active site of an enzyme) with an exterior solvent. The CAVER algorithms presented in this paper are active site of an enzyme) with an exterior solvent. The example of a tunnel is depicted in Figure 2. The minimum of radii of all empty balls forming the tunnel is called tunnel bottleneck radius. To compute these tunnels, we use the Voronoi diagram introduced in the following section.

### 1.3 Voronoi Diagrams

Let the atoms be represented by \( M = \{ M_1, \ldots, M_k \} \) – a set of balls \( M_i = (C_i, r_i) \) in \( \mathbb{R}^3 \) with centers \( C_i \) and radii \( r_i \). We assume that no ball is completely contained inside another ball, even though intersections between balls are allowed. Then, the signed distance of a point \( X \in \mathbb{R}^3 \) to a ball \( M_i \) is defined as \( d(X, M_i) = \text{dist}(X, C_i) - r_i \), where \( \text{dist} \) is the Euclidean distance. The Voronoi region for a ball \( M_i \) is the set of points

\[
VR_i = \{ X \in \mathbb{R}^3 | \forall M_j \in M, j \neq i : d(X, M_i) \leq d(X, M_j) \}.
\]

The additively weighted Voronoi diagram (AWVD) for \( M \) is the set of Voronoi regions \( \{ VR_1, \ldots, VR_n \} \). Furthermore, a point shared by four or more Voronoi regions is called Voronoi vertex and a curve shared by three Voronoi regions is called Voronoi edge [13]. If all the balls have the same radius, we speak about ordinary Voronoi diagram (VD), or Voronoi diagram of points, which are in this case the centers of the balls. More information about the properties of VDs and algorithms for their construction can be found in [13]–[16].

### 1.4 Related Work

In this section we present several existing techniques for tunnel computation.

Geometry-based tunnel identification algorithms can be used for several different purposes. First, a single pathway going through the whole structure

![Fig. 1: Tunnels \( T_1 \) and \( T_2 \) connecting buried site \( A \) in macromolecule \( M \) with an exterior space.](image1)

![Fig. 2: The tunnel connecting points \( A \) and \( B \). The dashed line represents the tunnel centerline, the blue discs denote the balls of the tunnel and the violet discs represent atoms from \( M \).](image2)
and connecting two surface sites can be analyzed. For this purpose, the tools HOLE [17] and PoreWalker [18] were developed.

Second, the aim can be to identify all putative pathways in a structure. This type of analysis can be performed by Chunnel [19], which performs topological analysis of the structure using a grid, or the tool presented by Lindow et al. [20] using AWVD. The tool simplifies the graph corresponding to AWVD by applying a series of filters. Then the paths connecting two user-specified points can be computed on the filtered graph. The utilization of AWVD for analysis of biomolecules was also proposed in [21], [22] and implemented in tools BetaTunnel, BetaVoid and BetaMol. Another implementation of AWVD is the awVoronoi project following the approach described in [23].

The third possible usage of the geometry-based algorithms is the detection of pathways connecting a user-specified buried site with an exterior space. The CAVER 3.0 tool described in this paper falls into this last category, along with CAVER 1.0 [4], [24], CAVER 2.0 [25], MOLE 1.2 [26], MOLE 2.0 [27] and MolAxis [28]. CAVER 1.0 finds tunnels using a grid, while CAVER 2.0, MOLE 1.2 and MOLE 2.0 use VD of atom centers. MolAxis models large atoms by multiple balls for better approximation of AWVD of atoms of different radii. However, there are only few tools and algorithmic approaches providing means to analyze these pathways in dynamical structures. So in the rest of this section we will describe the existing solutions to this task.

MOLE 1.2 [26] offers the most similar functionality to work presented in this paper. It uses clustering for finding the correspondence between tunnels from different conformations. The similarity of tunnels is computed by comparing the sets of atoms lining the tunnels, but no information about the clustering algorithm is available. Experiments revealed that the clustering depends on the ordering of the tunnels [9]. The clustering of tunnels performed by MOLE was not able to clearly separate tunnels into clusters corresponding to known transport pathways [9]. Furthermore, the identification of tunnels in MOLE 1.2 is based on the assumption that the differences in radii of different atoms are negligible and uses the ordinary VD of atom centers, which can lead to the underestimation of the tunnel bottleneck radius by as much as 50% of its actual value [9].

The suitable choice of the algorithm for the clustering of tunnels was investigated by Benes et al. [29]. The geometrical similarity of two tunnels was computed using the distance function derived from the Hausdorff distance. However, their clustering experiments were performed on hundreds of molecular conformations which is too limited with respect to the length of current simulations reaching up to tens of thousands of conformations. Tracing of the shape of a selected pathway from a single snapshot through time has been also described by Benes et al. [30]. Their method searches in the closest neighborhood of a single fixed pathway. However, it overlooks the second closest tunnel, which can be nearly identical to the initial tunnel, but wider than the tunnel actually found in a given snapshot. A completely different approach to geometry-based tunnel detection in dynamical structures was proposed by Benes et al. in 2011 [31]. The pathways are assembled from individual cavities, which were detected in the consecutive snapshots and are overlapping geometrically. A similar principle was also used to analyze and visualize dynamics of pathways and cavities by Lindow et al. [32], [33].

2 Algorithms

This section describes the algorithms implemented in the CAVER 3.02 tool for tunnel discovery and analysis. The essential inputs are a single structure or the ensemble of its conformations and the position of a buried site which should be connected with an exterior space via detected tunnels. The workflow of tunnel discovery first constructs the Voronoi diagram which is used for identification of all tunnels fulfilling the input parameters. These tunnels are then postprocessed and clustered in order to study their dynamics. The outputs of the process consist of the geometry of detected tunnels, their visualization, and their properties. The individual steps of the workflow will be described in the following sections.

2.1 Construction of the Voronoi Diagram

This section first focuses on the construction of the approximate AWVD using ordinary VD. Within this process a special case can occur which is described as well. Then the derivation of VD from the Delaunay triangulation is discussed.

2.1.1 Approximation of the Additively Weighted Voronoi Diagram

For the representation of a macromolecule, the AWVD is the most appropriate structure. The better availability of implementations of the ordinary VD for points led us to prefer approximation of the AWVD by ordinary VD.

The utilization of ordinary VD of atom centers leads to considerable error. To take into account that atoms have different radii and limit this error, we identify the smallest atom with the radius \( r \) and approximate all greater atoms by multiple balls with the radius \( r \).

All such atoms are approximated by 13 balls of radius \( r \). The icosahedron is placed to the center of the atom, 12 balls are centered at vertices of the icosahedron and one ball is placed at the centre of the atom. The size of the icosahedron is maximal such that all the 12 balls lie inside the atom. The
result for atoms commonly present in biomolecules is shown in Figure 3. The idea was inspired by the approximation approach used by MolAxis which is described in detail by Yaffe [34].

The maximum difference between the surface of the atom with the 1.8 Å radius (van der Waals radius of sulphur) and its approximation by balls of the 1.2 Å radius (van der Waals radius of hydrogen) is identical for icosahedron and dodecahedron. The difference is smaller than 0.18 Å. As the computational time of Voronoi diagram construction is almost half for icosahedron, we decided to use this platonic solid.

The advantages of our approach in comparison with approaches based on corrections of ordinary VD by shifting its planes [35] are the simplicity and the fact that the difference between the exact and the approximated atom surface is limited by a small constant. On the other hand, representing an atom by several balls results in higher computational time and memory costs. For the structure of the usual size of 7,000 atoms, the computation took 6 seconds and required 600 MB of RAM. For probably the greatest available structure with biological tunnel, the human ribosome containing over 230,000 atoms, the computation took 20 minutes and required 28 GB of RAM (tested on Intel Core i7-4960X 3.60GHz).

2.1.2 Special Case Precautions
When constructing the Voronoi diagram, usually no five centers of balls lie on a common empty sphere. Then each Voronoi vertex is connected by Voronoi edges to at most four other Voronoi vertices (see Figure 4 – in 2D, the vertices have three neighbors). Otherwise, a special case may occur and a Voronoi vertex can belong to more than four Voronoi edges. Due to limited accuracy of floating point numbers, even inexact special cases can be problematic. Some special case occurred in nearly every structure. Because dealing with special cases complicates both algorithms for Voronoi diagram construction and data structures for its representation, we implemented several precautions discussed below to make the occurrence of the special case practically negligible. After these precautions were implemented, no special case was detected in the test set of hundreds of structures.

Fig. 3: Atoms approximated by balls of equal radius. The first line depicts atoms of hydrogen, oxygen, nitrogen, carbon, sulfur and phosphorus. The second line shows their approximated representations by balls of radii corresponding to the hydrogen atom. The third line shows the approximation of the same atoms by the balls with radius of oxygen, which is the smallest atom in many available structures that do not include information about hydrogen atoms.

Fig. 4: a) Illustration of the 2D special case of a Voronoi vertex with four neighbors. b) If the disc $M_i$ representing an atom is moved so that its center is no longer co-circular with the centers of the other atoms, two vertices with three neighbors each replace the one with four neighbors.

The first precaution is the placement of a ball at the center of each icosahedron, which removes the special Voronoi vertex at this center. The second one is choosing a pseudorandom rotation of each platonic solid. This precaution also makes the error of the approximation independent on the choice of the coordinate system. As the third and most important precaution, the coordinates of each ball are changed by a small value, smaller than 0.001 Å (the coordinates in PDB files contain at most three digits after the decimal point). When computing VDs for molecular dynamics, the seed for generating the pseudo-random numbers is deterministically derived from the PDB identifier of each structure.

2.1.3 Triangulation
As a result of the previous step, we obtain a set of balls of equal radii as an input for VD construction. The Quickhull algorithm [36] is used to construct the Delaunay triangulation (DT). The VD is then constructed by exploiting its duality to the DT. Voronoi vertices are constructed for each tetrahedron as the centers of spheres circumscribed to the tetrahedra (see Figure 5). Every two Voronoi vertices from the neighboring tetrahedra are connected by a Voronoi edge. Each edge is represented by its vertices and by the empty ball that is chosen arbitrarily as one of the four balls nearest to the edge (e.g., the edge $E_1$ and the disc $M_i$ in Figure 5). This empty ball will be later used to estimate how far the points on the edge are from the neighboring atoms.
2.2 Geometrical Properties of a Voronoi Edge

Tunnels are identified as the cheapest paths in a VD graph. The cost of the path is defined as a sum of the costs of its edges. Let \( AB \) be the Voronoi edge connecting Voronoi vertices \( A \) and \( B \). The cost of the edge \( AB \) is defined as

\[
\text{cost}(AB) = \int_0^{\text{dist}(AB)} r(l)^{-z} \, dl.
\]

The value \( r(l) \) is the maximum radius of the ball of the tunnel that does not collide with atoms and which is centered on \( AB \) in the distance \( l \) from \( A \). The parameter \( z \) is a non-negative real number that allows users to choose the desired geometrical properties of tunnels. When \( z \) is set to 0 for all edges on the path, only the length of the path is taken into account and the shortest path is reported first. On the other hand, when \( z \) is set to its maximum allowed value of 100, the length of the path will become practically negligible in comparison with \( r(l) \), i.e., the wider paths will be preferred over the short ones. By default, the value is set empirically to \( z = 2 \) to give the priority to edges forming paths that are both wide and short [4, 26, 28].

The integral is enumerated using the trapezoidal rule with a uniform grid. During the process, for each edge the minimum \( r(l) \) is stored and later used as an estimate of the maximum radius of the empty ball that can travel along the edge without intersecting any atom. This minimum is called edge bottleneck radius.

2.3 Finding a Starting Vertex

Most usually, tunnels lead from/to the site where a chemical reaction can occur. Atoms are usually less densely packed in this site which allows the better fit of reactants. Users can then specify the position of this reaction site. However, this position often does not match with the ideal position of the reaction site which is both in the vicinity of the user specified site and as far from protein atoms as possible. This can lead to the underestimation of the radius of the tunnel in its start. In other words, the ideal position corresponds to the center of the maximal possible sphere which fits to the void space containing the user defined position. To detect the ideal position, we find the Voronoi vertex which fulfills the above stated two requirements. This vertex is then used as the first point of the tunnel centerline. The following algorithm is used to find this point.

As an input of the algorithm, the coordinates of the point \( S_{\text{initial}} \) have to be provided by the user, either directly or in the form of a set of atoms. In the latter case, \( S_{\text{initial}} \) is computed in each conformation separately as the centroid of the centers of the specified atoms.

Then let \( q_i \) be the maximal radius of the ball centered at a Voronoi vertex \( V_i \) such that the ball does not intersect any atom. The starting vertex \( S_{\text{start}} \) is identified as the Voronoi vertex \( V_i \) such that \( \text{dist}(V_i, S_{\text{initial}}) \) is smaller than the user-defined parameter \( d \) and \( q_i \) is larger than another user defined value, choosing the vertex closest to \( S_{\text{initial}} \) in case when more such vertices exist. If no such vertex exists, the same criteria are applied for value \( d = d_{\text{default}} \), which can happen when the user sets too small value of \( d \). If even such a vertex does not exist, the user-defined parameters are not used at all and the vertex closest to \( S_{\text{initial}} \) is chosen to be the starting vertex \( S_{\text{start}} \).

2.4 Vertices Stopping Search

The geometrical identification of transport pathways ceases to be meaningful on the interface between the macromolecule and the exterior solvent, where the geometrical constraint of space by atoms is no longer the major factor limiting the movement of small molecules. In this section, several sets of Voronoi vertices are defined, which are used to avoid the detection of tunnels in space that is opened to the exterior solvent.

First, Voronoi vertices with at most three neighbors in 3D, hereafter referred to as border vertices, are identified. For example, every vertex except for \( V_2 \) in Figure 5 (2D example) is a border vertex.

A set of vertices entirely excluded from tunnel computation is called outer vertices (see vertices marked with triangles in Figure 7). They are identified as those that can be reached by a spherical probe of radius \( r_S \) (default value 3 Å) from border vertices. This is achieved by the depth-first search performed on the graph composed of vertices and edges of VD starting from a set of border vertices.

The macromolecular surface contains many clefts that are open to the exterior space, but too narrow to be reached by the probe of radius \( r_S \). Consider the situation depicted in Figure 6. Tunnel centerline \( T_1 \) should not be considered a valid tunnel because it goes through a cleft that is opened to the exterior space. Furthermore, if the identification of this type

---

Fig. 5: Construction of Voronoi diagram from Delaunay triangulation. The dashed circle is circumscribed to the bold triangle and the Voronoi vertex \( V_i \) forms the center of this circle. \( E \) is an example of a Voronoi edge. The violet discs represent atoms.
of tunnels is not prevented, they would prevail in the results, making it difficult to separate them from relevant tunnels $T_1$, $T_2$, and $T_3$.

This motivates the introduction of a new set of vertices called the shallow vertices (see squared vertices in Figure 7) which is an extension of the set of outer vertices. The construction of the set of shallow vertices is described by Algorithm 1 and in the remainder of this paragraph. The neighborhood of each outer vertex $V$ is examined, and an empty ball of maximum radius such that it does not intersect with any atoms is centered at $V$. The ball radius is then increased by the user-specified value $depth$ (default 4 Å). Then, the space inside the enlarged ball is explored as follows. Each vertex that can be reached from $V$ by a spherical probe of radius $\rho_B$ (default 0.9 Å) is added to the set of shallow vertices.

Each vertex which is neither outer nor shallow is called inner vertex. The inner vertices (see vertices represented by circles in Figure 7) are located in the interior of the macromolecule, while the shallow vertices are closer to the exterior.

2.5 Tunnel Identification

Now we can proceed with the identification of tunnels in a single structure. Only Voronoi edges with radius greater or equal to the user-specified minimal tunnel bottleneck radius $\rho_B$ (default 0.9 Å) are used for that. We will use two graphs $(V, E)$ and $(W, F)$, where $V$ and $W$ are sets of vertices and $E$, $F$ are sets of edges. All four sets are defined below and illustrated in Figure 8.

$V$ is a set of vertices that are either inner vertices or connected to an inner vertex. $E$ represents a set of edges, where each edge connects two vertices of $V$ where at least one is an inner vertex. $W$ stands for a set of vertices that are either shallow but not outer, or outer and connected to a shallow vertex. Finally $F$ is a set of edges, where each edge connects two vertices of $W$ where at least one is not an outer vertex.

The shallow vertex connected by a Voronoi edge to at least one inner vertex is called shallow boundary vertex (double squared vertices in Figure 8). The outer vertex connected by a Voronoi edge to at least one shallow vertex is called outer boundary vertex (double triangled vertices in Figure 8).

The tunnels are then identified by a two step procedure. In the first step, the Dijkstra algorithm is used to find the lowest cost path (edges valued using equation 1) in graph $(V, E)$ from a starting vertex $S_{start}$ to every reachable shallow boundary vertex. In the second step, each path $H$ is extended separately – the Dijkstra algorithm is used to find the single lowest cost paths in the graph $(W, F)$ connecting the last vertex of $H$ with an outer boundary vertex. Then, only if an outer boundary vertex was reached, the path is considered to be a centerline of a tunnel.

The second step prevents reporting tunnel variants that branched from the tunnel of lower cost near to the protein surface. This can be viewed as an elaboration of the approach used in MolAxis [28], where the branching of tunnels beyond the sphere centered at the starting point is prohibited. The shapes of many structures are not spherical and the starting point is often far from the protein center. Thus there are many structures where users have no choice but to either miss the variants on one side of the structure or to be overwhelmed by many nearly identical tunnels on the other side, which are closer to the starting point. This is solved by our approach as it prohibits branching beyond the surface that follows the shape of the analyzed structure.

The Dijkstra algorithm cannot identify a path that
Algorithm 1: Identify Shallow Vertices

1. for $i$ from 0 to $g - 1$ do
2. \hspace{1em} Shallow[$i$] = false
3. end for
4. for $v \in$ Outer do
5. \hspace{1em} $Q \leftarrow$ empty FIFO queue
6. \hspace{1em} $Q$.enqueue($v$)
7. \hspace{1em} while $Q$ not empty do
8. \hspace{2em} $w \leftarrow Q$.dequeue()
9. \hspace{2em} Shallow[$w$] = true
10. \hspace{2em} for $i \in \{0, 1, 2, 3\}$ do
11. \hspace{3em} $x \leftarrow V[v][i]$
12. \hspace{3em} if Shallow[$x$] = false
13. \hspace{4em} and dist($v, x$) < $R_{\text{vertex}}[v] + \text{depth} - r_B$
14. \hspace{4em} and $r_B \leq R_{\text{edge}}[w][i]$ then
15. \hspace{5em} $Q$.enqueue($x$)
16. \hspace{10em} end if
17. \hspace{2em} end if
18. \hspace{1em} end for
19. \hspace{1em} end while
20. end for

joins a cheaper path before both paths reach shallow vertices. This so-called overshadowing problem has been previously described in \cite{28}. Such paths can be identified by manually increasing the depth parameter, but in the future, it would be desirable to develop a sufficiently efficient algorithm (with the execution time reaching from seconds to few minutes per structure) for the fully automated identification of all paths that are significantly dissimilar.

2.6 Tunnel Postprocessing

Each tunnel is now represented by the polyline composed of Voronoi edges which is called centerline. The centerline is transformed into the sequence of empty balls called the profile balls using the following procedure. Points are placed on the centerline in regular intervals and the maximal empty balls are placed at each point. Finally, the empty balls from the end of the tunnel are removed one by one until the empty ball with a radius smaller than or equal to $r_S$ (the radius of the probe for determining outer vertices) is reached. Thus, the center of the last empty ball of each tunnel lies in a practically negligible distance from the solvent accessible (Lee-Richards) surface of a protein structure determined by the probe of radius $r_S$ \cite{37, 38}.

2.7 Similarity of Tunnels

Two tunnels are considered to be similar if at least some portions of them lead through the same regions of the structure. Our measure of similarity of two tunnels is based on the Euclidean distance between pairs of points derived from the centerlines of the tunnels. This measure is used for two reasons. The first reason is to remove nearly identical tunnels within one conformation. The second is to find the correspondence between tunnels from different conformations by clustering (see Figure 9). This can be very time consuming because the evaluation of as many as $n \cdot (n - 1)/2$ distances can be required during the clustering phase, where $n$ is the number of all tunnels in all conformations. Therefore, the distance function should be calculated efficiently. It is also beneficial if the distance is a metric, because it can potentially be used to accelerate the clustering process. A distance function meeting both these requirements as well as the process of identification of points capturing the tunnel centerline geometry and the algorithm for the evaluation of the metric of tunnel similarity will be described in the following sections.

2.7.1 Representation of Tunnel Geometry

In order to evaluate the similarity between two tunnels efficiently, the centerline geometry of each tunnel is characterized by $h$ representative points (default $h = 10$). The distances between corresponding points are measured, i.e., when comparing tunnels $T$ and $U$, the Euclidean distances $\text{dist}(A_{T1}, A_{U1})$, $\text{dist}(A_{T2}, A_{U2})$, $\text{dist}(A_{T3}, A_{U3})$, $\text{dist}(A_{T4}, A_{U4})$, $\text{dist}(A_{T5}, A_{U5})$, $\text{dist}(A_{T6}, A_{U6})$, $\text{dist}(A_{T7}, A_{U7})$, $\text{dist}(A_{T8}, A_{U8})$, $\text{dist}(A_{T9}, A_{U9})$, $\text{dist}(A_{T10}, A_{U10})$.
the end of tunnel $T$. For this purpose, we designed the Algorithm 2, which is also described in the next paragraph.

First, a set of tunnel endpoints is computed, where each endpoint is identified as the point on the tunnel centerline lying in the furthest distance from $S_{\text{average}}$. Next, for each endpoint, we find all endpoints in its proximity. To make the search more efficient, we transform the set of all endpoints into a smaller set called aggregated points. This is performed using the procedure described in Algorithm 2. The purpose of the algorithm is to replace each group of points that are close together by a single point that is located reasonably near to their centroid. The points $A$ and $B$ are close if the angle $AS_{\text{average}}B$ is smaller than a threshold value. The weight of each representative point equals the number of points it represents. If the point being added is close enough to an already existing point, both points are merged using the function $\text{AV ERAGE}$. The direction and the distance of a new point is computed (Algorithm 2, line 5, 6). Both direction and distance take into account the weights $a$ and $b$ of the points merged, so that the new point lies closer to the point which represents more points.

The set of aggregated points is constructed iteratively by taking the endpoints one by one. The angles between the endpoint, $S_{\text{average}}$ and the aggregated points are considered. If the smallest angle is below the threshold value, the endpoint and the aggregated point are merged. The weighted average of their positions is used, where the weight of the aggregated point equals to the number points it represents. Otherwise, the endpoint is added to the aggregated point set.

For each tunnel centerline $T$ with endpoint $P$, a subset of aggregated points is identified such that each point $X_i$ from the subset must satisfy the condition that the angle $PSX_i$ is small enough (threshold value $5^\circ$). The value $r_T$ is then computed as the average of the distances of the points from this subset to $S$ (see the left side of Figure 10). Next, the sequence of spheres $K_1, \ldots, K_h$ (e.g., the spheres $K_{T1}, K_{T2}, K_{T3}$ in Figure 10), having the common center $S$ and radii $r_T/h, 2r_T/h, \ldots, r_T$ are constructed. Finally, each representative point $A_{T_i}, i \in \{1, \ldots, h\}$, is computed as the centroid of the intersection of the tunnel centerline and the space between spheres $K_{i-1}$ and $K_i$.

### 2.7.2 Metric for Evaluating Tunnel Similarity

The distance between tunnels $T$ and $U$ with sequences of representative points $A_{T1}, \ldots, A_{Th}$ and $A_{U1}, \ldots, A_{Uh}$ is defined as

$$\text{dist}(T,U) = \frac{1}{h} \sum_{i=1}^{h} \frac{k_2}{h-1} \text{dist}(A_{T_i}, A_{U_i}),$$

where $w$ is the linear function $w(x) = k_1 x + k_2$. The coefficients $k_1$ and $k_2$ are set so that the ratio between poor

... $\text{dist}(A_{Th}, A_{Uh})$ are evaluated (see Figure 10). The rest of this section describes how the representative points are computed and the computation of tunnel similarity from these points will be described in the following subsection.

Even a small change in position of surface atoms can cause significant changes in the solvent accessible surface and thus the length of a tunnel. To reduce the effect of these changes on the tunnel length when evaluating the tunnel-tunnel similarity, a sphere is assigned to each tunnel $T$. The tunnel will later be considered to end at the surface of this sphere for purpose of tunnel similarity estimation. The center of this sphere is positioned into the centroid of starting vertices from all conformations $S_{\text{average}}$. The radius of the sphere is derived by averaging the distances of $S_{\text{average}}$ and the tunnel ends in the vicinity of
Algorithm 2: Aggregate Tunnel Ends

Centerlines – set of tunnel centerlines (polylines)
S – abbreviation for centroid of starting vertices
$S_{\text{average}}$
$\alpha_{\text{max}}$ – threshold angle for point aggregation
Points – associative array, keys are unit vectors, values are weighted points
A, A_u, B, B_u, C, C_u, D – points; a, b – scalar values

1: function UNIT(A)
2: return A/|A|
3: end function

4: function AVERAGE(S, A, a, B, b)
5: $C \leftarrow a \cdot \text{UNIT}(A - S) + b \cdot \text{UNIT}(B - S)$
6: $D \leftarrow (a \cdot |AS| + b \cdot |BS|) \cdot \text{UNIT}(C)/(a + b)$
7: return $(S + D, a + b)$
8: end function

9: function AGGREGATE(Centerlines, S)
10: for $X \in$ Centerlines do
11: $A \leftarrow \text{point on } X \text{ most distant to } S$
12: $A_u = \text{UNIT}(A - S)$
13: if Points is empty then
14: Points.insert($A_u, (A, 1)$)
15: else
16: $(B, b) \leftarrow \text{Points.nearest}(A_u)$
17: $B_u = \text{UNIT}(B - S)$
18: if $a_{\text{max}} < \arccos(A_u \cdot B_u)$ then
19: Points.insert($A_u, (A, 1)$)
20: else
21: $(C, c) \leftarrow \text{AVERAGE}(S, A, 1, B, b)$
22: Points.delete($B_u$)
23: $C_u \leftarrow \text{UNIT}(C - S)$
24: Points.insert($C_u, (C, c)$)
25: end if
26: end if
27: end for
28: return Points
29: end function

The last point $w(1)$ and first point $w(0)$ is equal to the parameter $q$ (default 1), and $w(0.5)$ equals to 1. Setting $q$ to values smaller than 1 emphasizes the importance of the beginning of the tunnel, while values of $q$ larger than 1 emphasize the end of the tunnel.

This geometry-based metric is an alternative to the metric based on the comparison of sets of atoms lining the tunnel [26]. The geometric approach allows users to emphasize the importance of either end or beginning of tunnels for similarity estimation. On the other hand, the atom-based approach is more general, as the geometric approach assumes that the tunnel centerline continually increases its distance to the starting point, otherwise the ability of the metric to distinguish between dissimilar tunnels decreases.

2.8 Removal of Redundant Tunnels in One Conformation

Several highly similar tunnel centerlines can be identified in one static structure. To remove such redundant tunnels, the following iterative procedure is employed. The lowest cost tunnel $T$ is selected and all tunnels within the user-specified distance from $T$ are discarded. The procedure is repeated with the next remaining lowest cost tunnel, until all tunnels are either selected or discarded. The purpose of tunnel removal for each conformation separately is to reduce the data size in order to accelerate the subsequent clustering of all tunnels and to make the results of the computation on a static structure more comprehensible.

2.9 Tunnel Clustering

All remaining tunnels from all conformations are collected and clustered to allow the statistical analysis of the properties of corresponding tunnels, i.e., allowing to study the dynamics of tunnels. The lowest cost pathway of a cluster is selected from each conformation, providing the information about all changes of the locally most significant pathway in time.

2.9.1 Average Link Clustering

The number of tunnels, i.e., the elements to cluster, typically ranges from tens to hundreds of thousands and it can be expected to grow with the increasing length of molecular dynamics simulations [39]. Because tunnel identification in multiple conformations can be distributed among many computers, the clustering phase is the most important computational bottleneck in the workflow. In CAVER 3.0, we used the memory-constrained average link hierarchical clustering algorithm (C1) [9, 10] in the hope that it would allow for the efficient clustering of hundreds of thousands of tunnels, even with a computer not equipped with a large amount of RAM. However, its long computational time made clustering impractical. Therefore, the algorithm with the optimal worst-case computational complexity $O(n^2)$ was implemented instead (C2) according to the paper [11] and the corresponding source code from the author’s home page.

As $O(n^2)$ memory complexity imposes the limit for maximal size of the dataset due to the limited available RAM, we developed a modification of the algorithm using $O(n)$ memory and $O(n^3)$ worst case time complexity (C3). The difference is that instead of maintaining the cluster-cluster similarity matrix in the memory, the distances are computed on the fly. This slower algorithm is used until the number of clusters decreases to the value that allows the whole matrix to be stored in memory. From this point, the efficient $O(n^2)$ algorithm is used to finalize the cluster joining.
Algorithm 3: Preclustering
threshold – size of clusters specified by users
Tunnels – set of tunnels
dist – measure of tunnel similarity according to
\( \text{dist}(T, T) = 0, \text{dist}(T, U) = \text{dist}(U, T) \)
Preclustering – set of sets of tunnels

1: Preclustering ← {}  
2: \textbf{while} Tunnels not empty \textbf{do}  
3: \hspace{1em} \( T \) ← the lowest cost tunnel from Tunnels  
4: \hspace{1em} \textbf{for} \( U \in \text{Tunnels} \) \textbf{do}  
5: \hspace{2em} Cluster ← {}  
6: \hspace{2em} \textbf{if} \( \text{dist}(T, U) \leq \text{threshold} \) \textbf{then}  
7: \hspace{3em} remove \( U \) from Tunnels  
8: \hspace{3em} add \( U \) to Cluster  
9: \hspace{2em} \textbf{end if}  
10: \hspace{1em} \textbf{end for}  
11: add Cluster to Preclustering  
12: \textbf{end while}

2.9.2 Approximation

To allow for the clustering of very large datasets, two techniques for the reduction of the size of the data are used: subsampling and preclustering.

First, a user-defined percent of the tunnels (e.g., 20%) is randomly subsampled into set \( A \), whereas the remaining tunnels are placed into set \( B \). Tunnels in set \( A \) are preclustered using Algorithm 3. The resulting set of small clusters is then used as an input for the average link clustering, which aggregates these small clusters into a set of full-sized clusters. Then each tunnel in set \( B \) is assigned to one of the full-sized clusters using the Algorithm 4.

Algorithm 3 (preclustering), Algorithm 4 (subsampling) and average link clustering have the same asymptotic time complexity \( O(n^2) \). However, the subsampling is faster than the preclustering and average link clustering (the computational time is proportional to the subsample percentage). Both preclustering and subsampling have \( O(n) \) memory complexity. They reduce the amount of elements to be clustered by the average link clustering, thus overcoming its major limitation – high memory requirements. The limitation of subsampling is that it may destroy small clusters. However, this can be usually tolerated, because the biologically relevant tunnels appear in a large number of conformations.

The example of clusters obtained by the average link clustering and this approximation will be given in the following subsection.

2.9.3 Measurements

To illustrate how fast the tunnel clustering is in practice, we provide the computational times for the above mentioned algorithms. All of the times were obtained by clustering tunnels computed on 20,000 snapshots of molecular dynamics simulation of haloalkane dehalogenase enzyme DhaA [9], which is available at www.caver.cz. The default clustering threshold 3.5 Å was used (i.e., the average distance of tunnels in the cluster is smaller than 3.5 Å). In the case of algorithm \( C_3 \), the preclustering threshold 1.0 Å was used (i.e., each tunnel in the precluster is at most 1.0 Å far from the lowest cost tunnel in the precluster) and 20% of all tunnels were put into the set \( A \). All computations were performed by running CAKER 3.02 on Java OpenJDK Runtime Environment 1.7.0 with 14 GB maximum Java heap size on a computer equipped with an Intel Xeon E5-1620 v2 3.70 GHz processor and 16 GB RAM. The computational times are summarized in Table 1.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Time in mins</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 ) - old algorithm</td>
<td>576</td>
</tr>
<tr>
<td>( C_2 ) - all in quick phase</td>
<td>12</td>
</tr>
<tr>
<td>( C_2 ) - half in quick phase</td>
<td>38</td>
</tr>
<tr>
<td>( C_2 ) - all in slow phase</td>
<td>289</td>
</tr>
<tr>
<td>( C_3 ) - approximation</td>
<td>3.6</td>
</tr>
</tbody>
</table>

TABLE 1: Summary of computational times of clustering algorithms.

First, we focused on the comparison of algorithms \( C_1 \) and \( C_2 \) on 58,766 tunnels computed using the default bottleneck radius 0.9 Å. The clustering by the
algorithm \( C_1 \) previously used in CAVER 3.0 took 576 minutes. The algorithm \( C_2 \) working in the \( O(n^2) \) quick phase all the time finished in 12 minutes. When it was forced to work in \( O(n^3) \) phase all the time, it took 289 minutes. However, in practice, both phases are used. To test this, the quick phase was activated when the number of clusters dropped to half, which resulted in the computation time of 38 minutes. This setting has 1/4 memory cost compared to using only the quick phase. The computation time for the approximative algorithm \( C_3 \) was 3.6 minutes. However, the primary purpose of this algorithm is not to be faster than \( C_2 \), but rather to allow computations with \( O(n^2) \) time complexity in case there is not enough memory available to run \( C_2 \) on large data.

Second, we tested the quality of the clustering produced by the approximative algorithm \( C_3 \) by comparing it with the output of \( C_2 \). 99.6% of all tunnels are contained within the seven greatest clusters of the \( C_2 \) clustering. We compared each of those clusters with the most similar cluster from \( C_3 \) clustering, where the similarity of a pair of clusters is measured as the number of tunnels they share divided by the number of tunnels in the union of both clusters. The average cluster-cluster similarity was 93%. Six clusters were replicated very precisely (with greater than 96% similarity), but the similarity for the fifth greatest cluster was only 58%. The difference can be explained by the instability of average link clustering itself rather than by error of the approximation. We found out that the content of clusters varies significantly when different randomly selected subsets of tunnels are clustered by the average link algorithm. The similarity for the fifth cluster remained low (56%) even when just 10% of randomly sampled tunnels were removed. This suggests that the neighborhood of the fifth cluster cannot be unambiguously clustered at the given threshold and a different threshold value should be used for analysis of tunnels in this region.

Finally, we tested the capability of the approximation to process very large data by clustering 711,517 tunnels obtained using bottleneck radius 0.6 Å. The computation took 117 minutes.

The new clustering solutions allow the user to analyze dynamical tunnels faster and to process larger sets of tunnels. Thus, long molecular dynamics trajectories can be analyzed, which is essential for the observation of slow dynamical events (e.g., loop movements), while maintaining the level of detail essential to also monitor fast events (e.g., rotation of side chains) in such data.

### 2.10 Tunnel Properties and Visualization

Along with the tunnel calculation also several numerical characteristics of the tunnel geometries are calculated. Moreover, the tunnel can be visualized within the context of its molecule. These issues are discussed in this section.

There are several properties which can be computed and serve as another descriptor of the computed tunnel. After the previously described phases of the calculation, each tunnel is represented as a sequence of empty balls and belongs to a certain conformation and cluster. Typically, each conformation contains one or very few tunnels from each cluster. The smallest ball of the tunnel and its radius – the tunnel bottleneck – are frequently used values in tunnel analysis. It can be expected, however, that the suitability of the tunnel for transport of a small molecule is also influenced by its secondary bottlenecks and the tunnel length. Therefore, the cost (see equation [1] from section 2.2) of the tunnel should be a more appropriate predictor of tunnel relevance. However, because the distribution of costs for the evolution of one tunnel in time is asymmetric, the transformed value, called the throughput, is computed as \( e^{-cost} \), where \( e \) is the natural exponential function.

Clusters are then ordered by their priority, which is derived from the tunnel throughputs. The throughput in each cluster is averaged over all conformations, using zero value for conformations without any tunnel, and the greatest throughput value in conformations containing several tunnels.

The length of the tunnel is computed as \( \sum_{i=1}^{m-1} \text{dist}(Z_i, Z_{i+1}) \), where \( Z_1, Z_2, \ldots, Z_m \) is a sequence of the profile balls of the tunnel. A value called the curvature is computed as the ratio of the distance \( \text{dist}(Z_1, Z_m) \) and the tunnel length. Atoms lining the tunnel are computed as atoms within the user-specified distance from any ball of the tunnel. The tunnel and cluster properties are saved and can be further analyzed using a text viewer or spreadsheet editor.
2.10.2 Tunnel Visualization

There are several possibilities for the visualization of clusters and individual tunnels. To allow the comparison of the radii profiles of multiple tunnels, heat maps are generated. The heat map is an image representing values of a matrix of colors. Thus, the evolution of the radii profile of a whole tunnel in time can be captured in a single image, or an average tunnel radii profile can be constructed for each cluster allowing several different clusters to be compared.

Exporting the computed tunnels to existing molecular visualization applications allows for the visualization of tunnels in the context of their corresponding molecular system. Tunnels can be visualized as sequences of densely sampled spheres, or these spheres can be covered with a smooth surface to provide a cleaner visualization. These two representations are suitable for viewing the tunnels in individual conformations. To visualize clusters of tunnels from many conformations, showing each tunnel as a line representing its centerline provides more comprehensible visualization (see Figure 8). The ability of molecular visualization software PyMOL [43] and VMD [44] to display these representations is enabled in the exports of the CAVER results.

In all of these cases, tunnels are rendered using the same method as is used for molecular rendering, i.e., atoms are used to draw a profile ball and chemical bonds to draw a line. This approach is quite simple to implement and can be used to export the data to almost any molecular browser. On the other hand, this solution can cause performance problems when a large number of tunnels is visualized and the properties of tunnels cannot be displayed interactively. These limitations motivated the development of a novel software tool called CAVER Analyst [42]. The tool integrates CAVER and provides an easy to use graphical user interface for computation, comprehensive visualization (see Figure 11), and interactive exploration and evaluation of tunnels and their properties both on static and dynamic structures. Users can analyze tunnels interactively together with features such as heat maps or detailed information about the tunnel surroundings. Furthermore, it provides graphical methods suitable for the visualization of the shape of tunnels. CAVER Analyst introduces the asymmetric representation of the tunnel surface, which aims to represent the tunnel shape more precisely.

3 CONCLUSIONS AND FUTURE WORK

This paper introduces several algorithms for analysis of tunnels in biomacromolecules. Tunnels in individual structures are detected using the Voronoi diagram. Moreover, we introduce an algorithm for positioning the origin of tunnels into an empty space in the proximity of the user-specified point. Then, the algorithms for the demarcation of the surface in macromolecules are also presented. The metric for the efficient computation of the geometrical similarity of two tunnels is described. We utilize the clustering approach to find the correspondence between tunnels from different snapshots and to allow the analysis of changes of tunnel shape in time. An approximate clustering algorithm is available for processing of a large number of tunnels. Altogether, the algorithms allow the identification and analysis of tunnels in both static and dynamic structures.

Presented algorithms have some limitations, which can be addressed in future research. First, it must be noted that the geometry-based tunnel identification is approximate from its principle, since it does not consider the tunnel opening induced by the transported molecule, nor does it see energy barriers caused by other effects than the sterical clashes. However, to take into account the mentioned effects is significantly more time consuming, requires considerable experience and knowledge of the transported molecule. Therefore, geometry based methods are valuable complement to more complex simulations.

The inaccuracies caused by approximating larger atoms by multiple smaller balls can be expected to be smaller than the above mentioned limitations. However, we plan to utilize the additively weighted Voronoi diagram in the future to improve accuracy, efficiency, and the deterministic behavior of the algorithm.

The metric for tunnel-tunnel similarity can suffer from inaccuracy in distinguishing the parts of tunnels whose centerlines would follow a sphere centered at the starting point over large portion of length of the tunnels. It would be possible to measure the similarity of two tunnel centerlines using mutual minimal distances from one path to the other and vice versa, at the cost of the computational time. However, we did not implement such a solution, because the evaluation of tunnel similarity is a computational bottleneck and also because we have not encountered biologically relevant tunnels that would behave in the above mentioned way.

The previously mentioned overshadowing problem refers to the inability to identify multiple tunnels sharing a common exit. Even though almost all such tunnels could be identified by the repeated computations with different parameters, a more general and efficient solution would be desirable. We consider this to be the most significant limitation of the currently available algorithms.

The workflow contains two computational bottlenecks - Voronoi diagram construction and clustering. Voronoi diagram construction can be performed in parallel by distributing structures among several computers. In that case, computational time of Voronoi diagram construction and clustering becomes similar. Distributed version of clustering is probably not a good solution, because computational times are al-
ready within few hours for large data and running the computation on multiple computers would require additional effort from the users. However, the acceleration of the search for similar tunnels on lines 4 - 10 of Algorithm 3 by exploiting the properties of metric space could be useful.

Our primary motivation was to develop a tool for the analysis of tunnels in enzymes with narrow tunnels, because these narrow tunnels are hard to identify and analyze without a dedicated tool. However, our tool can be used for the detection of tunnels for different biological systems. It can be utilized even for the analysis of pores.

The program CAVER 3.02 and the academic version of CAVER Analyst 1.0 are freely available at www.caver.cz, together with the molecular dynamics simulation used for testing our new clustering solution and PyMOL session corresponding to Figures 9 and 11. This simulation can be useful also for the development and testing of any other algorithms and software tools for the analysis of molecular dynamic trajectories.

ACKNOWLEDGMENTS

The work of AP was supported by Ph.D. Talent Scholarship provided by Brno City Municipality. The work of JB was supported by the project Employment of Best Young Scientists for International Cooperation Empowerment (CZ.1.07/2.3.00/30.0037) co-financed from European Social Fund and the state budget of the Czech Republic. The Ministry of Education is acknowledged for financial support (LO1214, CZ.1.05/3.2.00/08.0144). MetaCentrum is acknowledged for providing access to their computing facilities, under the program Center CERIT scientific Cloud (CZ.1.05/3.2.00/08.0144).

REFERENCES


Antonín Pavelka received his MS degree in Bioinformatics from the Faculty of Informatics at Masaryk University in 2008 and is currently working there toward the PhD degree. He was awarded the Brno Phd Talent scholarship in 2011. His research interests include bioinformatics and machine learning.

Eva Sebestova received the PhD degree in environmental chemistry from the Faculty of Science, Masaryk University in 2011. Her main research interests include applied bioinformatics and rational design of proteins.

Barbora Kozlíková received the PhD degree in computer science from the Faculty of Informatics at Masaryk University in 2011. Currently, she is an assistant professor at the Department of Computer Graphics and Design at the same institution. Her research interests include visualization, computational geometry and bioinformatics.

Jan Brezovsky is a leader of theoretical team of Loschmidt Laboratories at the Department of Experimental Biology and the Research Centre for Toxic Compounds in the Environment, Masaryk University, Brno, Czech Republic. He received his PhD in Environmental Chemistry from Masaryk University in 2011. His main research topics are computational engineering of proteins, virtual screening and development of computational tools for protein engineering.

Jiří Sochor is an Associate Professor of computer science at the Faculty of Informatics at Masaryk University, Brno, The Czech Republic. His research interests include computer graphics, virtual reality and human-computer interaction. He received Ph.D. in Digital computers from Czech Technical University in 1981. He is a co-author of the book Modern Computer Graphics (in Czech).

Jiří Damborsky is the Josef Loschmidt Chair Professor of Chemistry and Professor of Biochemistry at the Faculty of Science at Masaryk University in Brno, Czech Republic. Research in his group is focused on protein and metabolic engineering. His group develops new concepts and software tools for protein engineering (CAVER, HOTSPOT WIZARD, PREDICTSNP), and uses them for the rational design of enzymes and bacteria with improved properties for biocatalysis, biodegradation and biosensing. He has published over 160 original articles, 14 book chapters, and has filed 5 international patents. He is a co-founder of the first biotechnology spin-off from Masaryk University Enantis Ltd.