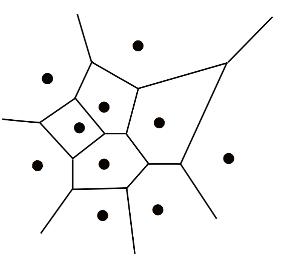
**Methods**

In short, a tunnel is a path between a point inside the macromolecular structure and its boundary. A tunnel is formed by its centerline and profile that specifies the tunnel radius in each of the centerline points. In essence, to find a tunnel the [previous version of MOLE](http://mole.upol.cz/about/) required the user to specify a starting point and from this point, the graph formed by the dual of the Delaunay triangulation (the Voronoi diagram, see below) got traversed in a depth first manner (paths with the lower cost were visited first). Each time a boundary vertex was reached, a tunnel got reported.

There are numerous problems with this approach. For example, when the first tunnel got reported, the next one was very similar to the first one (and therefore provided no useful additional information) because the path "branched" near the end point of the first tunnel. Another issue was caused by small "ridges" that were formed by vertices near the boundary of the diagram. As a result, a large part of the tunnel was often "going along" the surface of the protein and therefore didn't provide very relevant information about the structure of the tunnel. It was also very difficult to identify interesting starting points of the tunnels, unless prior knowledge.

The current version of the Mole 2 algorithm addresses these issues by preprocessing the Voronoi diagram, splitting it into several smaller parts (called **cavity diagrams**), and identifying suitable start- as well as end-points. In these cavities, suitable start and end points of the tunnels are identified and then Dijkstra's algorithm is used to find the tunnels.

**Mole 2 algorithm overview**

The tunnel computation in Mole 2 is performed in several steps:

1. Voronoi diagram representation of the protein is computed.
2. Diagram is split into several smaller parts – called cavity diagrams - that represent the empty space in the molecule. Start- and end-points are identified in each of the cavity diagrams.
3. Dijkstra's Shortest Path algorithm is used to find the tunnels between the pairs of starting and ending points.

**1. Macromolecular Voronoi Diagram Representation**

Voronoi diagram divides a metric space according to the distances between discrete sets of specified objects. In Mole case objects are the centers of the atom van der Waals spheres. The edges of the diagram represent the equidistant positions between pairs of closest atoms. The Voronoi diagram can be computed as a dual of the Delaunay triangulation of the vdW sphere centers. Therefore, the vertices of the diagram correspond to the tetrahedrons of the triangulation and are located at the tetrahedron's circumcenters. Similarly, the edges of the diagram represent the adjacency of the tetrahedrons.

**2. Preprocessing the Diagram**

The preprocessing works in several steps:

The boundary of the Voronoi diagram corresponds to the convex hull of the protein. This is not desirable because then the tunnel exits might end up being too far from the actual protein surface (in a way this is similar to the shallow ridge issue from the previous algorithm). To remedy this, the user is required to specify the **probe radius** parameter which is used to approximate the molecular surface. Given this parameter, layers of vertices are removed repeatedly starting from the boundary layer if the probe would pass through corresponding tetrahedron. This process is repeated as long as there is no longer any vertex to remove.

The second step is the removal of vertices of the Voronoi diagram (i.e. tetrahedrons) that cannot be part of any tunnel. This is governed by a parameter called the **interior threshold**. This parameter serves as an approximate lower bound on the tunnel radius. A vertex is removed if a sphere with the interior threshold radius cannot pass through any of the tetrahedron’s sides. After the boundary and interior vertices are removed, the **cavity diagrams** are computed as the connected components of the Voronoi diagram. In essence, these cavity diagrams represent the empty space inside the molecule. The last preprocessing step is to remove the shallow vertices. These are the vertices that form the above mentioned ridges along the surface of the molecule. All vertices with the distance (defined simply as the number of the vertices on the path) from the surface less than a given threshold (say 5) are removed if all their neighbors have the same or lower depth.

**3. Detection of Start points and End points**

Once the diagram is split into several smaller cavity diagrams, these are analyzed for suitable tunnel start- and end–points.. The general idea here is that the start point is the "deepest" vertex in the cavity and the endpoint is the "largest" (i.e. corresponding to the largest tetrahedron) boundary vertex.

**Start Points**

There are two ways to specify a start point for a tunnel - user specified list of residues or (Mole 2 only) automatic computation:

User defined:

The user specifies a list of residues or (Mole 2 only) a 3D point. Next, the centroid calculated from all the corresponding atomic centers is computed. Finally, for each cavity within a specified **origin radius** the closest vertex is selected as a start point.

Calculated: (Mole 2 only)

The topology of the cavity is used to calculate the start point. The calculated starting point is the "deepest" vertex of a cavity. The depth of a vertex is defined as the length of the path from this vertex to the closest boundary vertex.

**End Points**

Similarly to the start points, endpoints are either user defined (Mole 2 desktop version only) or calculated:

User defined:

These points can be specified by clicking on the surface of the molecule in the desktop application.

Calculated:

For each cavity diagram a subgraph B is induced by the boundary vertices. All vertices that would not fit a probe with radius defined by **opening threshold** parameter are removed. Then, the connected components of B are computed. Finally, for each component, the vertex with the highest volume is picked as a tunnel endpoint.

**4. Tunnel Computation**

Finally, when the set of start- and end-points is identified for each cavity, the Dijkstra's Shortest Path algorithm is used to find the tunnels between all pairs of start- and end-points using the weight function taking into account distances to the surface of the closest vdW spheres.

Then, the tunnel centerline is represented as a 3D natural spline defined by the Voronoi diagram vertices that form the path found by the Dijkstra’s algorithm.

**Final Note on the Complexity of the Algorithm**

The worst case complexity of the algorithm is O(M log M) where M is the number of vertices of the Voronoi diagram. In the worst case, M = N2 where N is the number of atoms. However, in most practical cases M is almost equal to N. Additonally, the complexity of calculating the Voronoi diagram is O(N log N). The complexity of all steps of the pre-processing phase is at most O(M log M). Finally, finding the paths using the Dijkstra's algorithm is O(K M log M) where K is number of tunnels. Together, the complexity of the algorithm is O(K N2 log N) where K is the number of found tunnels and N is the number of atoms in the molecule.