DOI: 10.1111/cgf.12980
Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

Volume 35 (2016), Number 5

Exploration of Empty Space among Spherical Obstacles via Additively Weighted Voronoi Diagram

M. Manak[1]

[1]NTIS – New Technologies for the Information Society, Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic

Abstract

Properties of granular materials or molecular structures are often studied on a simple geometric model - a set of 3D balls. If the balls simultaneously change in size by a constant speed, topological properties of the empty space outside all these balls may also change. Capturing the changes and their subsequent classification may reveal useful information about the model. This has already been solved for balls of the same size, but only an approximate solution has been reported for balls of different sizes. These solutions work on simplicial complexes derived from the dual structure of the ordinary Voronoi diagram of ball centers and use the mathematical concept of simplicial homology groups. If the balls have different radii, it is more appropriate to use the additively weighted Voronoi diagram (also known as the Apollonius diagram) instead of the ordinary diagram, but the dual structure is no longer a simplicial complex, so the previous approaches cannot be used directly. In this paper, a method is proposed to overcome this problem. The method works with Voronoi edges and vertices instead of the dual structure. Additional bridge edges are introduced to overcome disconnected cases. The output is a tree graph of events where cavities are created or merged during a simulated shrinking of the balls. This graph is then reorganized and filtered according to some criteria to get a more concise information about the development of the empty space in the model.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and geometric transformations

1. Introduction

A set of balls is a geometric model, which finds its use in many research areas. For instance, the balls can represent atoms in a molecular structure. Atom balls can have different radii and partially overlap. This simplification of a complex reality has been successfully used by scientists for years. The empty space outside atom balls is usually examined by a spherical probe of some predefined radius to locate cavities in enzymes or the empty tunnels through which a smaller molecule can reach the active site located in a cavity and trigger a chemical reaction there. Choosing a different probe radius may result in a different set of cavities, therefore probe variations are sometimes considered. Other use-cases include modeling of liquids, granular or porous materials, the growth of crystals, etc.

The geometric model in this paper is a finite set of balls that can have different radii. A probe radius can be added to all balls. Topological properties of the space outside all these balls may change during probe radius variations. This paper is focused on just one - the set of cavities. A method is proposed, which simulates the shrinking of the probe and creates a tree graph of cavities creation and merging. These events are detected on edges and vertices of the aw-Voronoi diagram (additively weighted). However, the resulting graph contains too much information. A hierarchical arrangement of cavities is proposed, which can be filtered according to user-defined criteria to get a more concise information. Capturing the changes in cavities during the shrinking of a spherical probes and their subsequent analysis may provide domain experts a better insight into the model. For instance in chemistry, experts will have a better chance to identify candidates to active sites of enzymes without specifying any probe radius.

The main idea is similar to previous methods of Delfinado and Edelsbrunner [DE93, ELZ02]. The key difference is that the previous methods are not applicable to the problem if the balls have different radii, as in the case of molecular structures because the methods are not able to model uniform expansion/shrinking of non-uniform balls. Other methods that detect cavities via aw-Voronoi diagrams or via their dual structures [KCKS13, KSS*14] require the probe radius to be specified prior to cavities detection, which is something that we really want to avoid in this paper.

This paper is organized as follows. Related work is briefly described in Section 2. Basic definitions of cavities and Voronoi diagrams are given in Section 3. The proposed method is described in Section 4 - first the concept of bridges, then the construction of the merge graph of cavities, and after that their hierarchy. Experiments and results are in Section 5 and the paper is concluded by Section 6.

2. Related Work

The situation is not complicated if all balls have equal radii and grow uniformly. In this case, the underlying structure is the ordinary Voronoi diagram of ball centers and its dual structure is a simplicial complex. All simplices in the complex can be sorted into a sequence, where each prefix represents an alpha complex. The sequence of successive prefixes is called the family of alpha complexes and it is an equivalent description of the system of growing balls with equal radii. All topological features of this system can be detected by the incremental algorithm of Delfinado and Edelsbrunner [DE93]. The algorithm can be conceptually divided into three parts. In the first part, the sorted sequence of simplices is processed from the beginning to the end, building components of 0- and 1-dimensional simplices (points and line segments). In the second part, the processing order is reversed and components of 3-dimensional simplices (tetrahedra) are built. Two 3-dimensional simplices are connected via a 2-dimensional simplex. Topological changes are detected via changes in the number of components and via detection of cycles. The simplices on which a topological change occurred are marked. At last, the sequence of simplices is processed once more to count topological features according to the marked simplices. The union-find system [GF64] is used to manage connected components of simplices.

The situation does not get much worse is the balls have different radii which do not change. In this case, the underlying structure is the Voronoi diagram using the power distance and the related concept of *weighted alpha-complexes* can be used [Ede92]. Edelsbrunner and Fu developed a method for measuring the volume and surface area of a union of balls and for the detection of cavities (voids) [EF94]. The dual structure forms a simplicial complex of tetrahedra, triangles, line segments and points. Simplices in the alpha-complex describe which pairs, triplets and quadruplets of the balls overlap, whereas remaining simplices describe the empty space outside all these balls. Components of the alpha-complex complement represent cavities. Later on, alpha-complexes have been used to measure cavities and pockets in proteins [EFFL95, LEF*98, LWE98]. Atoms in a protein are modeled as partially overlapping balls of different radii and a spherical probe of some predefined radius is used to inspect the empty space. Cavities are detected among the balls expanded by the probe radius.

Using weighted alpha complexes is problematic if probe radius variations are considered because the topology of the underlying Voronoi diagram may change. Nevertheless, weighted alpha complexes can still be used if only a few discrete variations of the probe radius are considered. An example is the work of Liang and Dill regarding the packing of proteins, liquids and crystalline solids [LD01]. A probe radius interval was sampled and the number of cavities was computed for each sample.

An interesting extension of [DE93] is the work of Edelsbrunner, Letscher and Zomorodian [ELZ02] regarding topological persistence and simplification. As the balls grow, topological features may appear and disappear, so the lifetime of a topological feature can be measured. This is used to distinguish important long-living features from the short-living ones. We should mention here that the method was designed for weighted alpha complexes, but if the balls have different radii, their growth is non-uniform.

For uniformly growing balls of different radii, the aw-Voronoi diagram can be used as an underlying structure. It is also known as the Apollonius diagram [BWY06]. Its regions have hyperbolic boundaries, so they are harder to compute.

The most widespread algorithm is the edge tracing algorithm of Kim et al. [KCK05], or a similar Voronoi S-network algorithm [LMOT99, MVLG06]. They both use the idea of tracing Voronoi edges to discover Voronoi vertices [TOO83]. The running time is quadratic on protein models. Spatial filters and parallelism can be used to speed it up [CKL*06, MK10, LBH11, OV14]. Furthermore, the Voronoi 1-skeleton (the union of edges) can have several components. Disconnected cases were solved recently [MK16].

Kim et al. call the dual structure of the aw-Voronoi diagram a *quasi-triangulation* [KKCS06, KCS10a] and they proposed the concept of beta-shapes and beta-complexes [KSK*06, KCS*10b]. A beta-complex uses a parameter β equal to the probe radius, which controls the size of balls. This allows to grow the balls by any probe radius and still have the description of overlaps. A weighted alpha-complex has a similar parameter α, which also controls the size of balls, but the balls of different sizes grow differently.

The aw-Voronoi diagram has also been used for the detection of tunnels and cavities in proteins [KCKS13], for the computation, visualization and analysis of Connolly surfaces [RKC*05, RPK07, MJK16] and molecular paths [LBH11]. The method [RKC*05] also navigates a shrinking probe along diagram edges, but it does not handle disconnected cases. The method [MJK16] extracts Connolly surface elements from the aw-Voronoi diagram. The analytical description of these elements is then uploaded to GPU and efficiently rendered via ray-casting. Interactive changes of the probe radius are possible. During this process, the creation and merging of cavities can be observed. Nevertheless, some user interaction in the form of changing the probe radius is required.

Yaffe and Halperin came up with an idea to approximate larger balls by a number of balls having the same size as the smallest ball [YH10]. Thanks to this approximation, they can still use the ordinary Voronoi diagram.

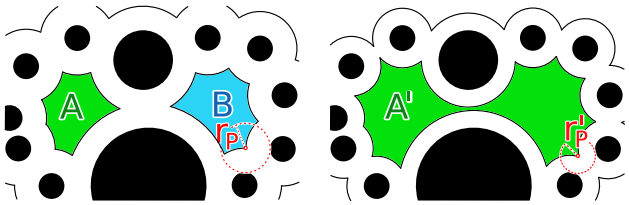
3. Computational Geometry Background

This section provides basic definitions and computational geometry background, which is necessary for the understanding of this paper. It includes the definition of cavities, aw-Voronoi diagrams, and the related concept of a Voronoi skeleton with extra structure.

3.1. Cavities

Let S be a set of balls and r_P be the probe radius. The space $\mathbb{R}^3 \setminus \bigcup_{s \in S} \{x \in \mathbb{R}^3 : ||x - center(s)|| < radius(s) + r_P\}$ may consist of one or more components. They will be called *cavities*. Exactly one cavity has infinite diameter.

Figure 1 illustrates cavities for two probe radii r_P and r'_P. The balls are enlarged by the probe radius and the complementary space of their union is classified into cavities A and B. As the probe radius decreases, both A and B will eventually merge into one, which is denoted as A' in Figure 1(b).



(a) Two different cavities A and B for the probe radius r_P

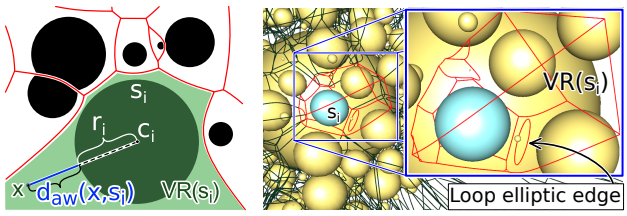
(b) A single cavity A' for the probe radius $r'_P < r_P$

Figure 1: *Balls and cavities. As the probe radius shrinks, both cavities A and B will eventually merge into one.*

If we detect the creation and merge events for a shrinking probe, we will be able to count cavities for any probe radius and to track the development of cavities w.r.t. the probe radius. These events are best detected if an appropriate Voronoi diagram is available for the balls. This diagram will be discussed in Section 3.2.

3.2. Additively Weighted Voronoi Diagram

The *aw-Voronoi diagram* (additively weighted or Apollonius diagram) is the set $VD(S) = \{VR(s_1), \ldots, VR(s_n)\}$ of Voronoi regions defined as $VR(s_i) = \{x \in \mathbb{R}^3 : \ \forall s_j \in S \ d_{aw}(x,s_i) \leq d_{aw}(x,s_j)\}$, where $d_{aw}(x,s_i) = ||x-c_i|| - r_i$ is the *signed aw-distance* of x from s_i. The aw-distance of $x \in \mathbb{R}^3$ from the whole set S is $d_{aw}(x) = min_{s \in S}(d_{aw}(x,s))$. Figure 2 shows some examples.



(a) 2D example

(b) 3D example

Figure 2: *Examples of aw-Voronoi diagrams*

The boundary of a 3D Voronoi region consists of k-dimensional elements: *faces* ($k = 2$), *edges* ($k = 1$) and *vertices* ($k = 0$). Such a k-dimensional element is a component of the intersection of some $4-k$ Voronoi regions. We will assume that the balls, to which these regions are assigned, are associated with the corresponding elements in an appropriate data structure, so we will be able to get the 4 balls for any vertex, the 3 balls for any edge, and the 2 balls for any face. Then we can also evaluate $d_{aw}(x)$ for any x lying on a Voronoi element as $d_{aw}(x,s_i)$ for any s_i assigned to the element.

The geometry of a face is a hyperboloid, the geometry of an edge is a conic curve, and the geometry of a vertex is a point or a pair of points. For each Voronoi element, the implicit function can be constructed from the corresponding set of balls, e.g., the hyperboloid of a face f with assigned balls $\{s_i, s_j\}$ is given by $d_{aw}(x,s_i) = d_{aw}(x,s_j)$.

Edges with the geometry of an ellipse will be called *elliptic edges*. Elliptic edges that are not incident to any Voronoi vertex will be called *loop edges*. The union of the geometric representation of all vertices and edges will be called a *Voronoi 1-skeleton*.

Non-linear Voronoi elements cause certain problems. One problem is that the same set of $4-k$ balls can be assigned to many k-dimensional elements, which means that many faces can share the same hyperboloid, many edges can share the same hyperbola, and two vertices can share the same pair of points. The other and more severe problem is disconnectedness of the Voronoi skeleton. Voronoi faces may contain holes and the boundary of each hole is formed by a cycle of one or more elliptic edges. These edges belong to a component of the skeleton, disjoint with the remaining edges and vertices in the boundary of the face. Figure 2(b) shows a 3D example of disconnected $VD(S)$. The loop edge causes a hole in a Voronoi face.

Aw-Voronoi diagrams can be computed by the edge tracing algorithm [KCK05, MVLG06]. Implementations are available, e.g., the proprietary QTFier [VDR16] or the open-source AwVoronoi [AWV16].

3.3. Voronoi Skeleton with Extra Structure

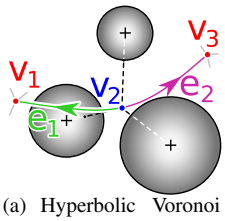
Possible movements of a shrinking probe can be described by a Voronoi diagram with extra structure, so-called enriched Voronoi diagram [Sie99, LBH11]. The extra structure is provided by including critical points of $d_{aw}(.)$ and orienting edges. For our purposes it is just a directed graph $\vec{G}$. It can be constructed as follows.

The Voronoi 1-skeleton is analyzed for *local extrema* of $d_{aw}(.)$ restricted to the Voronoi 1-skeleton. Extreme points of a Voronoi edge e lie in the intersection of e with the plane passing through the centers of three balls defining the geometry of e, provided by an appropriate data structure. Coordinates of extreme points can be computed by finding the center of a circle inscribed to three other circles, i.e., solving the Apollonius 10th problem. Only extrema lying on e are valid, but this can be decided with the angular distance comparison [KCK05]. The source code for computing extreme points can be found, e.g., in the open-source project AwVoronoi [AWV16].

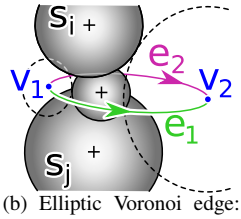
Vertices of $\vec{G}$ are exactly all Voronoi vertices together with all extreme points and a vertex at infinity $v_0 = v_\infty$. Edges of $\vec{G}$ are constructed by splitting Voronoi edges in extreme points and orienting the edges in the direction of increasing $d(.)$. All Voronoi edges leading to infinity are incident to v_0. Figure 3 shows an example of $\vec{G}$.

The orientation of edges is useful for the navigation of a shrinking probe. The probe center moves on edges and the probe radius equals to $d_{aw}(center)$. In each vertex of $\vec{G}$, edges point to vertices where the probe could possibly came from, with the exception of vertices representing maxima of $d_{aw}(.)$ on elliptic edges, e.g., as in the case of v_2 in Figure 3(b), where this information is missing.

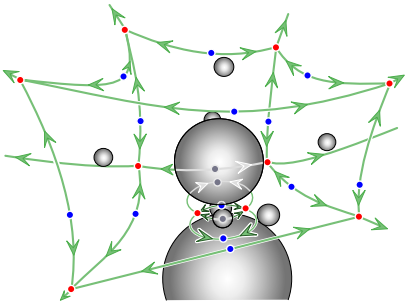
The output degree of any vertex is ≤ 2. It is a consequence of geometrical constraints that hold in Voronoi diagrams. It can be proven by examining the cases how a quadruple of balls can hold a probe and prevent the probe escape if the probe radius is locked and cannot change.



(a) Hyperbolic Voronoi edge: Voronoi vertices v_1, v_3, extreme vertex v_2 (minimum).



(b) Elliptic Voronoi edge: Extreme vertices v_1 (minimum) and v_2 (maximum).



(c) A more complex example of $\vec{G}$

Figure 3: *The vertices of $\vec{G}$ are the Voronoi vertices (red) together with extreme points (blue). Voronoi edges are split in these points and oriented in the direction of increasing $d_{aw}(.)$.*

4. Proposed Method

The description of the proposed method can be conceptually divided into subsections. First, a new concept of bridges is introduced to overcome certain difficulties with Voronoi diagrams. Next, a method is presented, which detects all cavities in the union of given balls for any probe radius. The resulting tree graph captures the behavior of cavities during contiguous shrinking of the probe radius. After that, this tree is transformed to get a more concise representation. At last, the time complexity is discussed.

4.1. Bridges

A new concept of bridges is proposed in this section. Bridges solve problems with elliptic Voronoi edges and they also solve disconnected cases by connecting one component of the Voronoi 1-skeleton to another. A *bridge* is an oriented connection between vertices of $\vec{G}$ with $d_{aw}(.)$ non-decreasing along the connection. For each vertex $v_m \in \vec{G}$ representing the local maximum of $d_{aw}(.)$ on an elliptic Voronoi edge, a bridge target can be computed by the following strategy.

Three balls defining the ellipse of v_m should be provided by an appropriate data structure. Among them, the pair s_i, s_j is chosen, which defines the Voronoi face domain $F(s_i,s_j) = \{x \in \mathbb{R}^3 \ : \ d_{aw}(x,s_i) = d_{aw}(x,s_j)\}$ in which the ellipse carves a hole. It is the pair s_i, s_j maximizing $||c_i - c_j|| + r_i + r_j$ because the third ball must be in the convex hull of s_i and s_j [Wil99]. After that we define a circle $\{x \in F(s_i,s_j) \ : \ d_{aw}(x,s_i) = d_{aw}(v_m,s_i)\}$. It lies in $F(s_i,s_j)$ and passes through v_m as the dotted circles in Figure 4 for $v_m \in \{v_1, v_2, v_4\}$. Then we compute intersection points of the circle and the edges of $\vec{G}$, find the edge with minimal non-zero Euclidean distance of the intersection from v_m, and choose the target vertex of the edge as the target point of the bridge. In Figure 4, $v_1 \to v_4$ and $v_2 \to v_3$ are such bridges. If there is no intersection, as in the case of $v_m = v_4$ in Figure 4, we search through the vertices of $\vec{G}$ in $F(s_i,s_j)$ for the vertex with minimal aw-distance greater than $d_{aw}(v_m,s_i)$ and use it as the target point. In Figure 4, v_5 is the target point. If no such point can be found, then the vertex at infinity is used as the target. For efficiency reasons, required groups of edges and vertices of $\vec{G}$ are collected in advance, mapped to the corresponding pairs s_i, s_j, and retrieved later when the intersection points need to be computed. Each group contains all edges/vertices in the boundary of all faces that lie on $F(s_i,s_j)$.

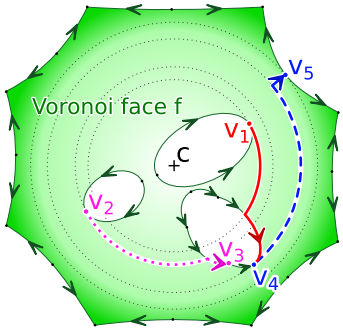


Figure 4: *Bridges $v_1 \to v_4$, $v_2 \to v_3$ and $v_4 \to v_5$*

Figure 5 illustrates bridges on a real example. The coordinates (x,y,z,r) of the balls are as follows: $(-2.5, 7.51, 0, 1)$ $(2.5, 7.51, 0, 1)$ $(0.01, 0.01, 80, 78)$ $(0, 0.01, -85, 78)$ $(0.5, 4, -10, 8)$ $(100.01, 0.01, 0.01, 80)$ $(0, 104, 0, 80)$ $(-100, 0, 0, 80)$ $(0, -100.01, 0, 80)$.

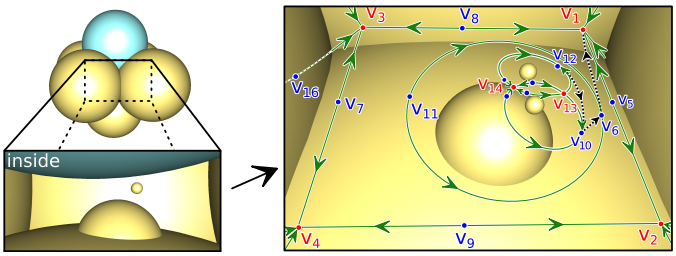


Figure 5: *The graph $\vec{G}$ was constructed from a Voronoi 1-skeleton. The bridges are $v_{12} \to v_{10}$, $v_{10} \to v_6$ and $v_6 \to v_1$.*

4.2. Finding All Possible Cavities

The method presented in this section finds all possible cavities in a given set of balls for all possible probe radii. If a probe radius is added to the radii of all balls, the components of the space outside all these balls represent cavities. The method simulates a contiguous shrinking of the probe radius. As the probe shrinks, new cavities are created or existing cavities merge. These events occur only on Voronoi vertices and edges. More precisely, in the local

extrema of $d_{aw}(.)$ restricted to the Voronoi 1-skeleton. The method processes the diagram and constructs a tree graph of the detected events. Nodes in the tree are organized - this will be further discussed in Section 4.3. The method is formalized in Algorithm 1. Its description follows.

First, the Voronoi graph $\vec{G}$ is created from the Voronoi 1-skeleton by splitting Voronoi edges in their extreme points and orienting them from low values of $d_{aw}(.)$ to higher values. This was discussed in Section 3.3. The aw-Voronoi diagram and the split points can be computed, e.g., by the open-source library AwVoronoi [AWV16].

Next, vertices of the graph are sorted in the non-increasing order of $d_{aw}(.)$. This is necessary to detect cavities in the right order.

After that, bridge targets are computed for the vertices of $\vec{G}$ that lie in the maxima of elliptic Voronoi edges. Details to the computation of bridges are in Section 4.1.

Next, some initialization is performed (line 6). For the sake of simplicity, we will assume only one vertex at infinity v_0. The first cavity is detected at v_0 as if the probe radius was ∞. Groups of graph vertices will be maintained. Each group corresponds to exactly one cavity (the cavity can be the result of several merges). The union-find system [GF64,CLRS09] will be used to manage groups, similarly as it was done in [DE93]. Each group will be realized by a tree stored in the array *parent*.

After the initialization, all vertices of $\vec{G}$ need to be processed. For each vertex v, the set N of its neighbors is obtained. These neighbors are the destination vertices of all outgoing edges of v. Several cases can occur. If the set N is empty, then $d_{aw}(v)$ is the local maximum of $d_{aw}(.)$ on the Voronoi 1-skeleton. The following two cases are distinguished from each other. If v is a Voronoi vertex (line 11), then a new cavity is created and a new group is started for v. Otherwise, v is necessarily the point of local maximum on an elliptic edge. Note that we already have bridges for all such points. In this case (line 14), v cannot trigger the creation of any cavity because any probe centered in v having a constant radius could escape along the bridge. Therefore, the group at v is simply inherited from the bridge target. Similarly, the group is also inherited if v is a Voronoi vertex with some neighbors (line 17). Thanks to the geometrical properties of aw-Voronoi diagrams, all the neighbors are already in the same group, so no merging is needed. Two cavities can merge only in points of local minima of $d_{aw}(.)$. These minima are on Voronoi edges, not in Voronoi vertices. This case is handled in the end (line 19). If v is the point of local minimum, it has two outgoing edges, i.e., two neighbors. The groups at these neighbors are located and merged. If the groups were not the same, it means that two cavities just merged into one, so a new merge node is created. Children of the node are ordered for the purpose of building the hierarchy of cavities in Section 4.3.

The output of Algorithm 1 can also be used to count cavities in any interval of probe radii. Each cavity creation event increases the number of cavities by one and each merge event decreases the number by one. The events just need to be ordered by their creation time (the value of the probe radius at which the event occurred). The infinite cavity should not be counted by default, i.e., both the initial and the final number of cavities is zero.

Algorithm 1: Find Cavities

Input : The set S of balls and the aw-Voronoi diagram $VD(S)$
Output: The root of the tree with cavities creation & merging

```
1  G⃗ = the Voronoi graph from VD(S);
2  V = vertices of G⃗, sorted: i ≤ j ⇒ d_aw(V[i]) ≥ d_aw(V[j]);
3  for v ∈ V do
4      if isEllipticEdgeMaximum(v) then
5          bridgeTarget[v] = getTarget(v); // Sec. 4.1
6  v_0 = V[0]; parent[v_0] = v_0; result[v_0] = root = new Cavity(v_0);
7  size[root] = ∞;
8  for i = 1 to |V| − 1 do
9      v = V[i]; N = neighbors of v in G⃗; // 0 ≤ |N| ≤ 2
10     if N == ∅ then // local maximum in v
11         if isVoronoiVertex(v) then
12             result[v] = root = new Cavity(v);
13             parent[v] = v; size[root] = d_aw(v);
14         else // elliptic edge maximum in v
15             parent[v] = parent[bridgeTarget[v]];
16         end
17     else if isVoronoiVertex(v) then
18         parent[v] = parent[any member of N];
19     else // local minimum in v ⇒ merge
20         R = {find(n, parent) : n ∈ N}; // 1 ≤ |R| ≤ 2
21         parent[v] = union(R, parent);
22         if |R| == 2 then
23             l = result[R[0]]; r = result[R[1]];
24             if size[l] < size[r] then
25                 swap(l, r); // Ensures size[l] ≥ size[r]
26             result[parent[v]] = root = new Merge(v, l, r);
27             size[root] = size[l];
28         end
29     end
30 end
31 return root;
32 function union({a, b}, p)
33     return (rank(a) < rank(b)) ? (p[a] = b) : (p[b] = a);
34 function find(n, p)
35     return (n == p[n]) ? n : (p[n] = find(p[n], p));
   // See [GF64, CLRS09] for union-find
   // Remaining functions are trivial
```

4.3. Cavities Hierarchy

The output of Algorithm 1 is a tree graph, which describes how cavities were created and how they merged during the shrinking of the probe used for their detection. This graph is often very deep and not very informative, at least in molecular models. A hierarchy proposed in this section aims to provide a better representation.

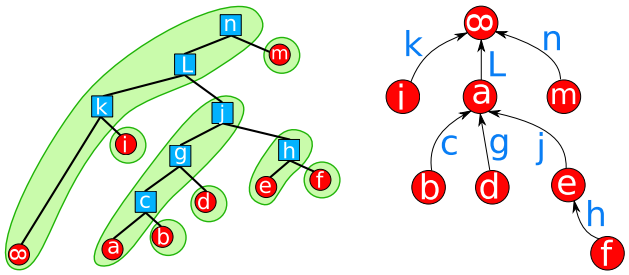
The hierarchy is based on a simple rule: If two cavities merge, then the cavity that was detected by a smaller probe dies by merging into the cavity detected by a larger probe. In the case of a tie, the choice can be arbitrary. This way, the life of a cavity can be tracked from its creation, i.e., from a leaf node up to the root until the cavity dies by merging with another. This path is transformed to a single

node in the hierarchy. Since the graph produced by Algorithm 1 is already organized, right sub-trees always merge into left sub-trees. The hierarchy is then constructed by Algorithm 2. An example of the transformation is shown in Figure 6.

Algorithm 2: Build Hierarchy

Input : The *root* of the tree with cavities creation & merging
Output: The hierarchy of cavities

```
1  v0 = leftmostLeaf(root);
2  result.add(new Node(v0));
3  push(stack, root);
4  while stack is not empty do
5      v = pop(stack);
6      if v is not a leaf then
7          lr = leftmostLeaf(rightChild(v));
8          lv = leftmostLeaf(v);
9          result.add(new Node(lv));
10         result.add(new Link(lr → lv, v));
11         push(stack, leftChild(v));
12         push(stack, rightChild(v));
13     end
14 end
15 return result;
   // push(), pop(), leftmostLeaf() and
   // rightChild() are easy to implement
```



(a) The tree of cavities creation (leaves) and merging (inner nodes)

(b) The hierarchy

Figure 6: *Creation of the hierarchy of cavities*

The hierarchy is a tree rooted in the cavity of infinite size. Each child in the hierarchy was born after its parent and dies before its parent by merging into the parent. Therefore, we can consider parents to be representatives of their children. In this interpretation, the level of detail can be changed by hiding or showing children.

However, there is a catch. Not all children of a parent are so important. For instance, the parent can have a large number of tiny short-living children (less important) and a few of large long-living children (more important). Some sort of filtering based on cavity properties is necessary. Some useful properties of cavities in the hierarchy may include the radius of the probe used to detect the cavity, the radius of the probe for which the cavity died by merging, the lifetime of the cavity as the difference between these two radii, the depth in the hierarchy tree, etc.

The decision about importance of cavities should be let on users by providing them a convenient way to define filters and specify thresholds. In the experimental implementation of the proposed methods, the hierarchy was exported to a graph and the Gephi software tool [BHJ09] was used for visualization and filtering.

We should mention here, that a similar concept called topological persistence and simplification was proposed by Edelsbrunner et al. [ELZ02], but the lifetime of a cavity is measured differently. Perhaps the lifetime proposed by Yaffe et al. [YH10] is more close to the approach proposed in this paper.

4.4. Time Complexity

The topology of the Voronoi diagram can be directly downloaded for most molecular structures from the VDRC website [VDR16] or computed by the edge tracing algorithm [KCK05, MVLG06].

The edge tracing algorithm has the following time complexity. Let n be the number of balls. The diagram has $O(n^2)$ edges in the worst case [Aur87, BK03] and the edge tracing algorithm spends $O(n)$ time on each of them, which gives $O(n^3)$ in the worst case. The estimate is more optimistic on real data such as protein models: The diagram should have $O(n)$ edges. The algorithm should also spend much less time on every edge if spatial filters are used [CKL*06, MK10, OV14], ideally $O(1)$ on average, but it depends on the efficiency of the filters.

In Algorithm 1, the graph $\vec{G}$ is constructed from the Voronoi diagram in linear time w.r.t. the number of Voronoi vertices and edges. All Voronoi vertices will become $\vec{G}$ vertices. Each Voronoi edge is analyzed for extreme points in $O(1)$. Voronoi edges are split in the extreme points and will become graph edges.

To construct a bridge, all edges in the boundary of all affected faces are examined. The worst case time complexity is quadratic w.r.t. the number of such edges. This is not very efficient in theory, but these cases are extremely rare in practice and the number of such edges is often very low, ideally $O(1)$.

The rest of Algorithm 1 has its time complexity dominated by sorting the vertices of $\vec{G}$, i.e., $O(|V|\,log|V|)$. The merging of groups can be done with the union-find algorithm [GF64, CLRS09], which has the amortized cost per operation $O(\alpha(n))$, where $\alpha(n)$ is the inverse of the fast growing Ackermann function; $\alpha(n)$ is often considered $O(1)$ for any practical value of n.

The time complexity of Algorithm 2 is linear if results of $leftmostLeaf(.)$ are precomputed during the construction of inner nodes of a merge graph.

5. Experiments and Results

The proposed methods were tested on sets of balls that produce disconnected aw-Voronoi diagrams and on models of molecular structures. The data of many molecular structures can be downloaded from public databases via their PDB ID.

Table 1 illustrates real running time on a few PDB files. The implementation is in Java and it was tested on CPU Intel i7 3.4 GHz (6x2 threads). $|V|$ is the number of Voronoi vertices. T_{VD} is the time

taken by AwVoronoi [AWV16] to compute the quasi-triangulation. $T_{\vec{G}}$ is the transformation time of the quasi-triangulation to the Voronoi graph $\vec{G}$, including bridges (only one in 1AON), i.e., lines 1-5 in Algorithm 1. T_{CAV} is the time of finding cavities in $\vec{G}$, i.e., lines 6-31. T_H is the time of hierarchy construction. A reasonable expectation is that the whole computation for small or medium-size proteins will take units of seconds and for large molecular structures tens of seconds. The last row in Table 1 is a special dataset of 10K points generated randomly in a cylinder with the radius 10 and the height 0.2 and two spheres with the radius 500, centered in $(0,\ 0,\ 500.1)$ and $(0,\ 0,\ -500.1)$. AwVoronoi [AWV16] was inefficient on this dataset. The diagram is shown in Figure 5. It has 292 components (142 loop edges). 2118 bridges were created.

Table 1: *Performance. K - thousands, s - seconds.*

Data	Atoms	|V|	T_{VD}	$T_{\vec{G}}$	T_{CAV}	T_H
1GKI	20K	133K	1.9s	0.5s	0.2s	0.1s
1AON	60K	400K	5.6s	1.6s	0.4s	0.1s
1JJ2	90K	620K	8.0s	2.4s	0.7s	0.2s
70S_RF2	300K	2060K	31.0s	9.0s	2.0s	0.4s
Random	10K	35K	25.1s	2.9s	0.1s	0.1s

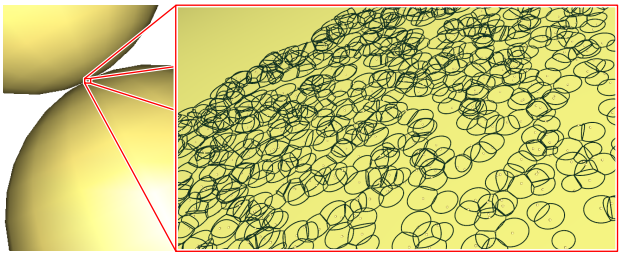


Figure 7: *Random data with many elliptic edges*

An interesting property is the dependency of the number of cavities on the probe radius. The results of the proposed method were checked against an alternative method, which uses weighted alpha shapes for cavities detection [LEF*98] and samples the probe radius interval by a constant step.

Figure 8 shows the results of one of such experiments. It is the set of balls from Figure 5. The configuration of balls was carefully chosen to get a short-living cavity in the vertex v_{14}. The proposed method detected all cavities in this setting of balls without any problem. Bridges were used to propagate the information about cavities in disconnected cases. The results were checked by counting cavities with the method based on weighted alpha shapes, but the interval of probe radii had to be sampled very densely to detect also the short-living cavity in the probe radius interval $(2,3)$.

The proposed method has been compared to other approaches, which specialize on the detection of cavities for a fixed probe radius. Results are shown in Figure 9. With these approaches, the only way to get the graph of the number of cavities w.r.t. the probe radius is to sample a probe interval and get the result for each sample. The values obtained in these samples should be the same as in the case of the proposed method. BetaMol [CKR*12] uses the aw-Voronoi diagram of atoms to extract cavities [KCKS13]. Voroprot [OMV11]

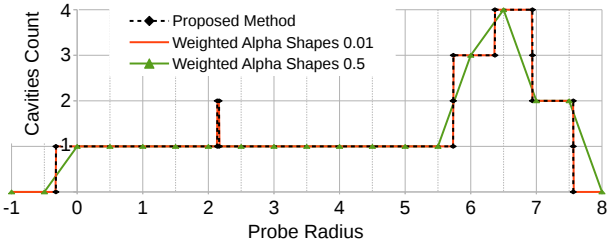


Figure 8: *The proposed method detected all cavities in a set of balls that caused disconnected aw-Voronoi diagram. The method based on weighted alpha shapes missed one cavity when the probe interval was regularly sampled using the step size* 0.5. *The cavity was detected when the step was set to* 0.01.

also uses the aw-Voronoi diagram, but cavities are defined differently, so the results do not match. The results of an approach based on weighted alpha shapes [LEF*98] is also shown. With a dense probe axis sampling step 0.01, the alpha-shapes method has results very close to the proposed method. However, the whole alpha-shape needs to be fully recomputed and analyzed for each sample, since the topology of the underlying structure may change for different probe radii. All these recomputations may take quite a long time and, in the end, some cavities can still be missing.

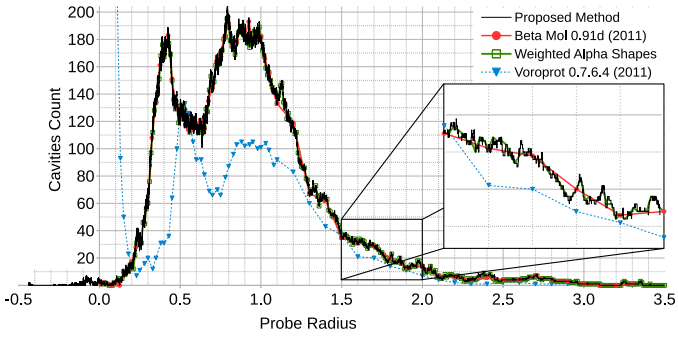
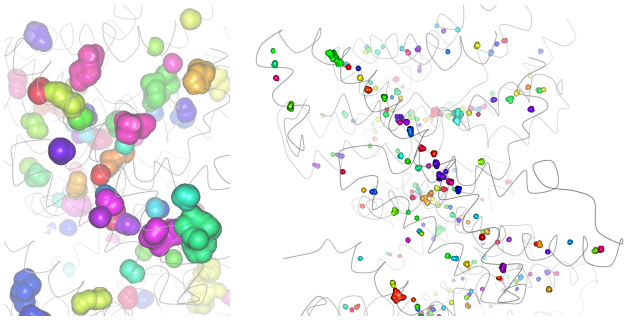


Figure 9: *Comparison of the proposed method with Beta-Mol [CKR*12], Voroprot [OMV11] and the method based on alpha-shapes [LEF*98]. Voroprot gives different results because it defines cavities differently. PDB ID: 4P1T (5075 atoms).*

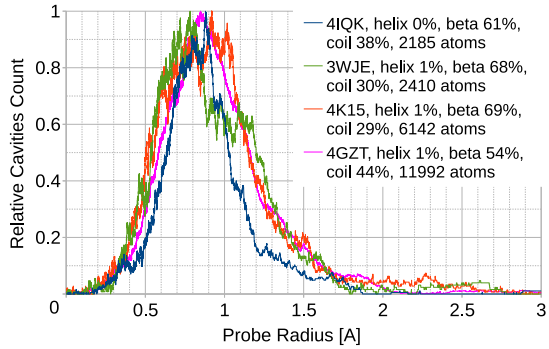
More experiments on molecular structures revealed that the number of cavities depends on the percentage of helix formations. A small probe can be trapped by atoms in a helix formation and create a micro-cavity. The periodic nature of helices results in a high number of such micro-cavities. Normal cavities are shown in Figure 10(a) whereas micro-cavities in Figure 10(b). Atom balls are hidden in these figures, but the backbone of the molecular structure is shown to highlight helix formations. Cavities are visualized by their Connolly surface and distinguished by different colors.

Figure 11 shows the difference between molecular structures having a low percentage of helices and structures having a high percentage of helices. The peek around the probe radius 0.35 in Figure 11(b) is caused by helix formations.

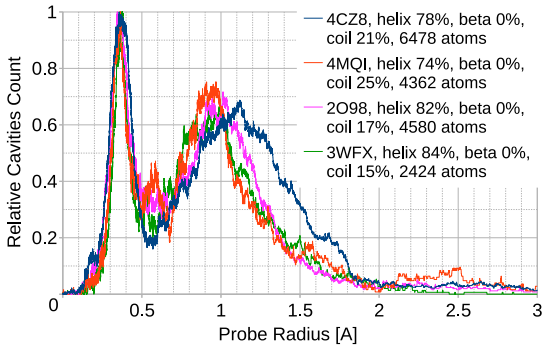


(a) Cavities for $r_P = 1.4$Å (b) Micro cavities for $r_P = 0.35$

Figure 10: *Cavities detected and rendered by CAVER Analyst [KSS*14] for the probe radius r_P. PDB ID: 4CZ8.*



(a) Low percentage of helices



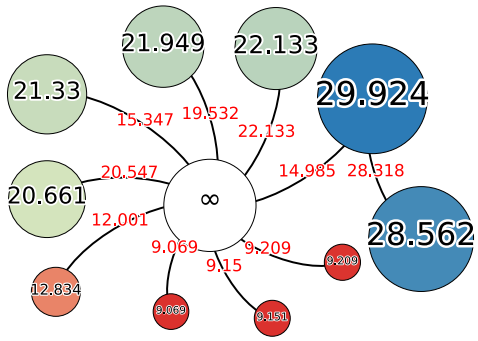
(b) High percentage of helices

Figure 11: *Cavities count in molecular structures*

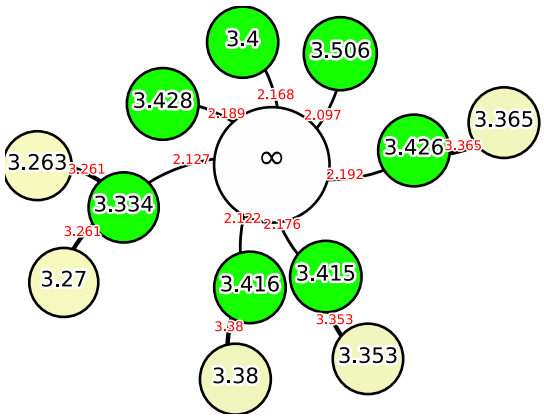
The rest of this section demonstrates the advantages of the proposed hierarchy of cavities. The following examples were computed by the implementation of the proposed method, converted to a directed graph and rendered by the Gephi software tool [BHJ09]. Gephi allows to bind the visual appearance of nodes to their attributes, filter the graph interactively, define layouts, etc. In the following graphs, nodes are labeled by the birth attribute (the radius of the largest probe that detected the cavity) and edges are labeled by the merge attribute (the radius of the probe for which the cavity merged into its parent). Numbers are rounded to 3 decimal places.

The first structure on which we will demonstrate the hierarchy of cavities is a large chaperonin complex with PDB ID: 1AON (58 674 atoms). The hierarchy has 69 930 nodes and maximal depth 6. However, 99.45% of all nodes have their depth ≤ 3, and 95.81% have their depth ≤ 2, and 75.92% have their depth ≤ 1. We are usually interested in nodes of depth 1 because they represent the cavities that will directly merge with the infinite cavity. The number of nodes on this level is still too high. Some filtering is needed.

Figure 12(a) shows the hierarchy of cavities filtered to keep only the nodes born ≥ 9Å. This filter reduces the hierarchy to 0.02% of the total number of nodes. This hierarchy graph tells us that 10 cavities were born in the probe interval $[9, \infty)$, but only 9 of them merge directly with the cavity representing infinity. The four large cavities born at 20.661, 21.33, 21.949 and 29.924 are shown in Figure 13. They are visualized by their Connolly surface and distinguished from each other by different colors. The cavity 22.133 is not shown in Figure 13. It would be visible if the probe radius in CAVER Analyst was set to, e.g., $r_P = 22.1332$. Figure 12(b) shows the hierarchy for 1AON filtered to keep only the cavities born in the interval $[3\text{Å}, 5\text{Å}]$ and with a lifetime ≥ 1.11Å. The seven nodes on the level 1 of the hierarchy correspond to the cavities where ADP molecules are located (the PDB model has ADP molecules in these places). If the lifetime filter was not used, there would be many more cavities.



(a) Only the cavities born ≥ 9Å



(b) Only the cavities with a strong lifetime, born between $2-5$Å

Figure 12: *PDB ID: 1AON. The filtered hierarchy of cavities. Processed with Gephi [BHJ09].*

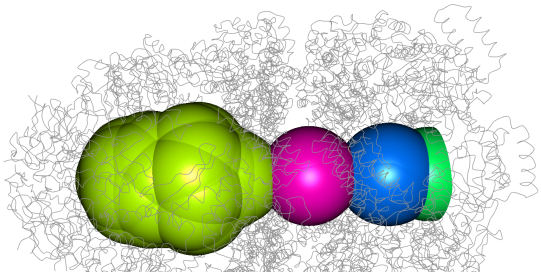


Figure 13: *PDB ID: 1AON. Four large cavities, probe radius* 20.66Å. *Rendered by CAVER Analyst [KSS* 14].*

The next structure on which we will demonstrate the hierarchy of cavities is an enzyme with PDB ID: 1CQW (2 358 atoms). The hierarchy has 2 947 nodes. The maximum depth is 6, but 97.73% of all nodes have their depth ≤ 3, and 92.53% have their depth ≤ 2, and 70.65% have their depth ≤ 1. Figure 14 illustrates how complicated the situation is without filtering. Figure 15 shows the hierarchy filtered by a composite filter. The filter restricts the cavities directly connected with ∞ to the ones that were born ≥ 1.4Å, died ≥ 1.0Å, and lived for ≥ 0.2Å . In other words, we are interested in cavities, which can accommodate at least one water molecule, are not entirely closed, and are relatively stable when the probe expands or shrinks a little bit. Only 0.98% nodes survived. The node 2.31 corresponds to the active site of the enzyme. The cavity is born at 2.31Å and dies at 1.477Å. If we had used a standard approach with the fixed probe radius 1.4Å, we would have missed the cavity.

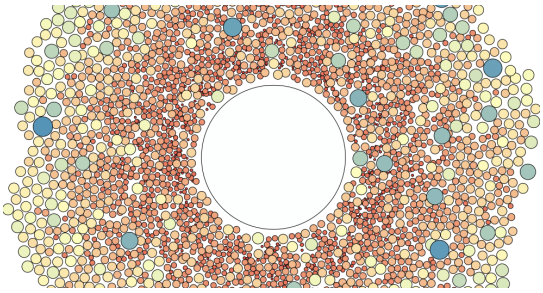


Figure 14: *PDB ID: 1CQW. The hierarchy without filtering.*

6. Conclusion

The number of cavities among balls depends on the radius of a probe used to inspect the empty space. The proposed method simulates a contiguous probe shrinking process, during which the empty space reachable by the probe is explored, and the behavior of cavities is recorded. The probe center moves along aw-Voronoi edges. Newly proposed bridges overcome problems with disconnected cases. The proposed hierarchy of cavities provides a better insight into the structure of the empty space among the balls than traditional approaches, which require the probe radius to be specified prior to cavities detection. The intended application is the analysis of bio-molecules, but the method is not limited to molecular data.

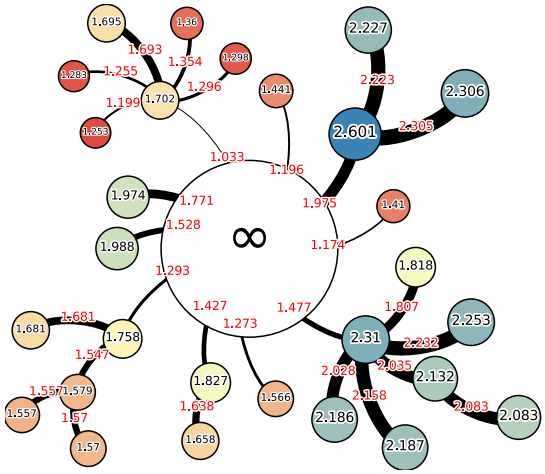


Figure 15: *PDB ID: 1CQW. The filtered hierarchy of cavities. Nodes are labeled with the birth radius (the greatest sphere that fits in the cavity), edges are labeled with the merge radius. The node* 2.31 *corresponds to the active site. Processed with Gephi [BHJ09].*

Acknowledgements

This publication was supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports.

References

[Aur87] AURENHAMMER F.: Power diagrams: Properties, algorithms and applications. *SIAM Journal on Computing 16*, 1 (1987), 78–96. doi:10.1137/0216006. 6

[AWV16] Computational library AwVoronoi, http://awvoronoi.sf.net, 2016. Accessed on May 20, 2016. 3, 5, 6, 7

[BHJ09] BASTIAN M., HEYMANN S., JACOMY M.: Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media* (San Jose, California, 2009), AAAI Press, pp. 361–362. 6, 8, 9

[BK03] BOISSONNAT J.-D., KARAVELAS M. I.: On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (Baltimore, Maryland, USA, 2003), SODA'03, ACM/SIAM, pp. 305–312. 6

[BWY06] BOISSONNAT J.-D., WORMSER C., YVINEC M.: *Effective Computational Geometry for Curves and Surfaces*. Springer Berlin Heidelberg, 2006, ch. Curved Voronoi Diagrams, pp. 67–116. doi:10.1007/978-3-540-33259-6_2. 2

[CKL*06] CHO Y., KIM D., LEE H.-C., PARK J. Y., KIM D.-S.: Reduction of the search space in the edge-tracing algorithm for the Voronoi diagram of 3D balls. In *Computational Science and Its Applications – ICCSA 2006, Proceedings, Part I* (Glasgow, UK, 2006), Gavrilova M., Gervasi O., Kumar V., Tan C., Taniar D., Laganá A., Mun Y., Choo H., (Eds.), vol. 3980 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 111–120. doi:10.1007/11751540_13. 2, 6

[CKR*12] CHO Y., KIM J.-K., RYU J., WON C.-I., KIM C.-M., KIM D., KIM D.-S.: BetaMol: A molecular modeling, analysis and visualization software based on the beta-complex and the quasi-triangulation. *Journal of Advanced Mechanical Design, Systems, and Manufacturing 6*, 3 (2012), 389–403. doi:10.1299/jamdsm.6.389. 7

[CLRS09] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN

C.: *Introduction to Algorithms (Third edition)*. MIT Press, 2009, ch. 21: Data structures for Disjoint Sets, pp. 561–585. 5, 6

[DE93] DELFINADO C. J. A., EDELSBRUNNER H.: An incremental algorithm for Betti numbers of simplicial complexes. In *Proceedings of the ninth annual symposium on Computational Geometry* (San Diego, California, USA, 1993), SCG'93, ACM, pp. 232–239. doi:10.1145/160985.161140. 1, 2, 5

[Ede92] EDELSBRUNNER H.: *Weighted Alpha Shapes*. Tech. rep., University of Illinois at Urbana-Champaign, IL, USA, 1992. UIUCDCS-R-92-1760. 2

[EF94] EDELSBRUNNER H., FU P.: *Measuring space filling diagrams and voids*. Tech. Rep. UIUC-BI-MB-94-01, Beckman Inst., Molecular Biophysics Group, University of Illinois at Urbana-Champaign, 1994. 2

[EFFL95] EDELSBRUNNER H., FACELLO M., FU P., LIANG J.: Measuring proteins and voids in proteins. In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences* (Wailea, HI, 1995), vol. 5 of *HICSS'95*, IEEE Computer Society, pp. 256–264. doi:10.1109/HICSS.1995.375331. 2

[ELZ02] EDELSBRUNNER, LETSCHER, ZOMORODIAN: Topological persistence and simplification. *Discrete & Computational Geometry 28*, 4 (2002), 511–533. doi:10.1007/s00454-002-2885-2. 1, 2, 6

[GF64] GALLER B. A., FISHER M. J.: An improved equivalence algorithm. *Communications of the ACM 7*, 5 (1964), 301–303. doi:10.1145/364099.364331. 2, 5, 6

[KCK05] KIM D.-S., CHO Y., KIM D.: Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. *Computer-Aided Design 37*, 13 (2005), 1412–1424. doi:10.1016/j.cad.2005.02.013. 2, 3, 6

[KCKS13] KIM D.-S., CHO Y., KIM J.-K., SUGIHARA K.: Tunnels and voids in molecules via Voronoi diagrams and beta-complexes. In *Transactions on Computational Science XX*, Gavrilova M. L., Tan C. J. K., Kalantari B., (Eds.), vol. 8110 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 92–111. doi:10.1007/978-3-642-41905-8_7. 1, 2, 7

[KCS10a] KIM D.-S., CHO Y., SUGIHARA K.: Quasi-worlds and quasi-operators on quasi-triangulations. *Computer-Aided Design 42*, 10 (2010), 874–888. doi:10.1016/j.cad.2010.06.002. 2

[KCS*10b] KIM D.-S., CHO Y., SUGIHARA K., RYU J., KIM D.: Three-dimensional beta-shapes and beta-complexes via quasi-triangulation. *Computer-Aided Design 42*, 10 (2010), 911–929. doi:10.1016/j.cad.2010.06.004. 2

[KKCS06] KIM D.-S., KIM D., CHO Y., SUGIHARA K.: Quasi-triangulation and interworld data structure in three dimensions. *Computer-Aided Design 38*, 7 (2006), 808–819. doi:10.1016/j.cad.2006.04.008. 2

[KSK*06] KIM D.-S., SEO J., KIM D., RYU J., CHO C.-H.: Three-dimensional beta shapes. *Computer-Aided Design 38*, 11 (2006), 1179–1191. doi:10.1016/j.cad.2006.07.002. 2

[KSS*14] KOZLIKOVA B., SEBESTOVA E., SUSTR V., BREZOVSKY J., STRNAD O., DANIEL L., BEDNAR D., PAVELKA A., MANAK M., BEZDEKA M., BENES P., KOTRY M., GORA A., DAMBORSKY J., SOCHOR J.: CAVER Analyst 1.0: graphic tool for interactive visualization and analysis of tunnels and channels in protein structures. *Bioinformatics 30*, 18 (2014), 2684–2685. doi:10.1093/bioinformatics/btu364. 1, 8, 9

[LBH11] LINDOW N., BAUM D., HEGE H.-C.: Voronoi-based extraction and visualization of molecular paths. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2025–2034. doi:10.1109/TVCG.2011.259. 2, 3

[LD01] LIANG J., DILL K. A.: Are proteins well-packed? *Biophysical Journal 81*, 2 (2001), 751–766. doi:10.1016/S0006-3495(01)75739-6. 2

[LEF*98] LIANG J., EDELSBRUNNER H., FU P., SUDHAKAR P. V., SUBRAMANIAM S.: Analytical shape computation of macromolecules: II. Inaccessible cavities in proteins. *Proteins: Structure, Function, and Bioinformatics 33*, 1 (1998), 18–29. doi:10.1002/(SICI)1097-0134(19981001)33:1<18::AID-PROT2>3.0.CO;2-H. 2, 7

[LMOT99] LUCHNIKOV V. A., MEDVEDEV N. N., OGER L., TROADEC J.-P.: Voronoi-Delaunay analysis of voids in systems of nonspherical particles. *Phys. Rev. E 59*, 6 (1999), 7205–7212. doi:10.1103/PhysRevE.59.7205. 2

[LWE98] LIANG J., WOODWARD C., EDELSBRUNNER H.: Anatomy of protein pockets and cavities: Measurement of binding site geometry and implications for ligand design. *Protein Science 7*, 9 (1998), 1884–1897. doi:10.1002/pro.5560070905. 2

[MJK16] MANAK M., JIRKOVSKY L., KOLINGEROVA I.: Interactive analysis of Connolly surfaces for various probes. *Computer Graphics Forum* (2016). doi:10.1111/cgf.12870. 2

[MK10] MANAK M., KOLINGEROVA I.: Fast discovery of Voronoi vertices in the construction of Voronoi diagram of 3D balls. In *Seventh International Symposium on Voronoi Diagrams in Science and Engineering* (Laval University, Quebec, QC, Canada, 2010), Mostafavi M. A., (Ed.), ISVD'10, IEEE Computer Society, pp. 95–104. doi:10.1109/ISVD.2010.22. 2, 6

[MK16] MANAK M., KOLINGEROVA I.: Extension of the edge tracing algorithm to disconnected voronoi skeletons. *Information Processing Letters 116*, 2 (2016), 85–92. doi:10.1016/j.ipl.2015.09.017. 2

[MVLG06] MEDVEDEV N. N., VOLOSHIN V. P., LUCHNIKOV V. A., GAVRILOVA M. L.: An algorithm for three-dimensional Voronoi S-network. *Journal of Computational Chemistry 27*, 14 (2006), 1676–1692. doi:10.1002/jcc.20484. 2, 3, 6

[OMV11] OLECHNOVIČ K., MARGELEVIČIUS M., VENCLOVAS Č.: Voroprot: an interactive tool for the analysis and visualization of complex geometric features of protein structure. *Bioinformatics 27*, 5 (2011), 723–724. doi:10.1093/bioinformatics/btq720. 7

[OV14] OLECHNOVIČ K., VENCLOVAS Č.: Voronota: A fast and reliable tool for computing the vertices of the Voronoi diagram of atomic balls. *Journal of Computational Chemistry 35*, 8 (2014), 672–681. doi:10.1002/jcc.23538. 2, 6

[RKC*05] RYU J., KIM D., CHO Y., PARK R., KIM D.-S.: Computation of molecular surface using Euclidean Voronoi diagram. *Computer-Aided Design and Applications 2*, 1-4 (2005), 439–448. doi:10.1080/16864360.2005.10738393. 2

[RPK07] RYU J., PARK R., KIM D.-S.: Molecular surfaces on proteins via beta shapes. *Computer-Aided Design 39*, 12 (2007), 1042–1057. doi:10.1016/j.cad.2006.10.008. 2

[Sie99] SIERSMA D.: Voronoi diagrams and Morse theory of the distance function. In *Geometry in Present Day Science* (University of Aarhus, Denmark, 1999), Barndorff-Nielsen O. E., Jensen E. B. V., (Eds.), World Scientific, pp. 187–208. 3

[TOO83] TANEMURA M., OGAWA T., OGITA N.: A new algorithm for three-dimensional Voronoi tessellation. *Journal of Computational Physics 51*, 2 (1983), 191–207. doi:10.1016/0021-9991(83)90087-6. 2

[VDR16] Voronoi diagram research center, http://voronoi.hanyang.ac.kr, 2016. Accessed on May 20, 2016. 3, 6

[Wil99] WILL H. M.: *Computation of additively weighted Voronoi cells for applications in molecular biology*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999. doi:10.3929/ethz-a-003845562. 4

[YH10] YAFFE E., HALPERIN D.: Approximating the pathway axis and the persistence diagrams for a collection of balls in 3-space. *Discrete & Computational Geometry 44*, 3 (2010), 660–685. doi:10.1007/s00454-009-9240-9. 2, 6