Practical and efficient computation of additively weighted Voronoi cells for applications in molecular biology

Author(s):
Will, Hans-Martin

Publication Date:
1998

Permanent Link:
https://doi.org/10.3929/ethz-a-006652878

Rights / License:
In Copyright - Non-Commercial Use Permitted
Practical and efficient computation of additively weighted Voronoi cells for applications in molecular biology

Hans-Martin Will

May 27, 1998

Abstract. This paper is concerned with the efficient computation of additively weighted Voronoi cells for applications in molecular biology. We propose a projection map for the representation of these cells leading to a surprising insight into their geometry. We present a randomized algorithm computing one such cell amidst $n$ other spheres in expected time $O(n^2 \log n)$. Since the best known upper bound on the complexity such a cell is $O(n^2)$, this is optimal up to a logarithmic factor. However, the experimentally observed behavior of the complexity of these cells is linear in $n$. In this case our algorithm performs the task in expected time $O(n \log^2 n)$. A variant of this algorithm was implemented and performs well on problem instances from molecular biology.

Author’s address:

Hans-Martin Will
Institute for Theoretical Computer Science
ETH Zentrum
CH-8092 Zurich
Switzerland

Technical Report #300, Departement Informatik, ETH Zürich.
Electronically available from:
1 Introduction

Consider a set of spheres \( S = \{\sigma_i, 1 \leq i \leq n\} \), each sphere \( \sigma_i = (c_i, r_i) \) defined by its center \( c_i \) and radius \( r_i \). If we denote by \( d(\cdot, \cdot) \) the Euclidean metric, then we can introduce the distance function \( d(x, \sigma) = d(x, c_i) - r_i \). For a point \( x \) outside \( \sigma \) this function measures the distance to the surface of the sphere. Finally, we can assign to each of the spheres \( \sigma_i \in S \) the set of all points "nearer" to \( \sigma_i \) than to all other spheres by defining

\[
V_i = V(\sigma_i) = \{x \in \mathbb{R}^3 : d(x, \sigma_i) \leq d(x, \sigma_j) \forall 1 \leq j \leq n\}.
\]

The set \( V_i \) is called the additively weighted Voronoi cell of the sphere \( \sigma_i \), the collection of all the cells \( V = \{V_i, 1 \leq i \leq n\} \) the additively weighted Voronoi tesselation induced by \( S \). We will also refer to \( V_i \) as the cell defined by \( \sigma_i \). This paper aims for the efficient computation of these cells.

1.1 Applications in molecular biology

Additively weighted Voronoi cells have been proposed as a local measure of the packing density of large macro-molecules such as proteins and nucleic acids. This density, in turn, is used to derive entropy-related energy terms or as a screening method for the verification of experimentally gained crystal structures. For an extensive discussion of the advantages of additively weighted Voronoi diagrams compared to other spatial tesselations see the recent paper [10]. For further discussions see [8, 22].

1.2 Previous work

All previous attempts to actually compute spatial additively weighted Voronoi cells for applications in molecular biology discretized the problem [9, 11, 15, 26], or they tried to adjust the vertices of the Voronoi cells of the centers of the spheres [27]. Aurenhammer observed that a cell of the additively weighted Voronoi diagram in \( \mathbb{R}^d \) can be represented as the projection of the intersection of a cell of a suitably defined power diagram in \( \mathbb{R}^{d+1} \) with a \( d+1 \)-dimensional cone [2]. If we use an optimal algorithm for computing convex hulls [6], we obtain an \( O(n^2) \) algorithm for computing the implicit representation of one or all \( n \) additively weighted Voronoi cells in \( \mathbb{R}^3 \). In practice, however, it turns out to be quite complicated to actually extract an explicit description of the cells from this implicit representation, and we are not aware of any complete implementation of this algorithm in \( \mathbb{R}^3 \). Therefore, we sought for an approach suitable for an actual implementation. Finally, there are general methods for computing lower envelopes of algebraic surfaces. In fact, our algorithm is based on ideas taken from [5, 4, 20]. These algorithms have running times — depending on the type of the geometric objects involved — between \( O(n^2 \log n) \) and \( O(n^{2+\varepsilon}) \), for arbitrarily small positive \( \varepsilon \). All these algorithms...
make use of a vertical decomposition scheme. Since it was reported that using vertical decompositions of spherical planar maps in the context of molecular modelling leads to numerical problems [13], we decided to search for a data structure which does not distinguish a specific direction. Again, we are not aware of an implementation of any of these algorithms for non-linear input objects.

1.3 Outline of paper

We begin our presentation by recalling some characteristic properties of additively weighted Voronoi cells and propose a projection map for their representation. With the new insights we gain from this representation, we are prepared to formulate an incremental algorithm for the computation of these cells. Besides theoretical considerations, we will outline the setup in which this algorithm is used for the intended applications. We conclude with first experiences obtained from an implementation.

2 Geometric properties

The geometric properties of additively weighted Voronoi cells can be found in [3, 18, 21]: A non-empty cell is a star-shaped region, and the center of its defining sphere is contained in the (visibility) kernel. The faces are hyperbolic surface patches and the edges are elliptic or hyperbolic arcs, which may degenerate to parabolic arcs or line segments. The latter is, for example, the case when all spheres have equal radii. In this case the additively weighted Voronoi diagram is just the ordinary Voronoi diagram of the centers of the spheres. If the spheres defining two adjacent cells have different radii, then the hyperbolic bisector separating these cells bends around the smaller sphere; in fact, the center of the smaller sphere is the focus of this hyperboloid. Therefore, a cell can be bounded without having a single vertex, as depicted in the left image of figure 1. This implies, that it does not suffice to concentrate on calculating the vertices of a cell. Indeed, as shown in the right image of figure 1, even the skeleton of the cell, i.e. the incidence graph of all vertices and edges is not always connected. Because of this, an algorithm really has to account for vertices, edges, and faces.

Since a non-empty cell is a star-shaped region, it is a natural approach to represent its surface using a spherical parameterization. We chose to parameterize a cell \( V \) by a unit sphere around the center \( c_i \) of its defining sphere \( \sigma_i \). Observe, that the diagram does not change if we add a constant \( \Delta r \) to the radii of all spheres in \( S \). To simplify the following presentation, we consider a cell \( V \) defined by a sphere \( \sigma \in S \) of radius 0 around the origin.

Let \( \pi : p \rightarrow \frac{p}{||p||} \) denote the map projecting a point \( p \in f, f \) a face of \( V \), onto the parameter range \( S^2 \). The collection \( P := \{ \pi(f) : f \text{ face of } V \} \) is a subdivision of \( S^2 \). For any element \( x \in P \) let \( \phi(x) \) denote its lifting back to the current boundary of the cell. As it is common parlance, we identify a map with its image.
Figure 1: Two special cases of additively weighted Voronoi cells. The left picture shows the cell of a small sphere between two larger ones having only a single elliptic edge. In the right picture, we added another small sphere to obtain a disconnected edge skeleton.

**Lemma 1** The projection of an edge in $V(\sigma) \cap V(\sigma_i) \cap V(\sigma_j)$ of an additively weighted Voronoi cell $V(\sigma)$ onto a unit sphere around the center $c$ of $\sigma$ is a circular arc.

**Proof:** Let $f$ be a face of $V$ that is defined by a neighboring sphere $\sigma_i = (c_i, r_i)$. Then the lifting map $\phi_i$ such that $\phi_i(p) = x$ is given by

$$\phi_i : p \mapsto \frac{c_i^2 - r_i^2}{2(r_i + \langle p, c_i \rangle)} \cdot p.$$  

We obtain this mapping by plugging the equations $x = d \cdot p$ and $\|p\| = 1$ into the equation of the graph of the distance function $(d + r_i)^2 = (x - c_i)^2$, where $d = d(x, \sigma_i)$. Equating the distance equations

$$d = \frac{c_i^2 - r_i^2}{2(r_i + \langle p, c_i \rangle)} = \frac{c_j^2 - r_j^2}{2(r_j + \langle p, c_j \rangle)}$$

for two neighboring spheres $\sigma_i = (c_i, r_i)$ and $\sigma_j = (c_j, r_j), 1 \leq i, j \leq n, i \neq j$ yields the equation describing the projection of $\phi_i \cap \phi_j$:

$$\langle a_{i,j}, p \rangle = b_{i,j}, \quad \text{where}$$

$$a_{i,j} = c_i \cdot (c_j^2 - r_j^2) - c_j \cdot (c_i^2 - r_i^2)$$

$$b_{i,j} = r_i \cdot (r_j^2 - c_j^2) - r_j \cdot (r_i^2 - c_i^2)$$

We denote the halfspaces defined in this way by $h_{i,j} = \{ x \in \mathbb{R}^3 : \langle a_{i,j}, x \rangle \geq b_{i,j} \}$. In the same manner, the boundary of the projection of a bisecting hyperboloid $V(\sigma) \cap V(\sigma_i)$...
is a circle on $S^2$, namely the circle defined by the vanishing points of the denominator of the lifting map $\phi_i$ satisfying $\langle p, c_i \rangle = -r_i$. We set $h_{0,0} = \{x \in \mathbb{R}^3 : \langle c_i, x \rangle \leq -r_i\}$, $h_{0,i} = \{x \in \mathbb{R}^3 : \langle c_i, x \rangle \geq -r_i\}$. To simplify notation, we define a symbolic lifting map $\phi_0 : p \to \infty$. For an illustration of this representation see figure 2.

3 An incremental algorithm

Let $\tau(n)$ denote a function bounding the expected complexity of a single additively weighted Voronoi cell $V(\sigma)$ amidst $n$ other spheres. The best known upper bound on the worst case complexity of such a cell is $O(n^2)$ [2]. On the other hand, the interior of a protein possesses a packing density comparable to crystal structures [16]. Probabilistic models of crystals and quasi-crystals based on additively weighted Voronoi cells suggest a constant bound on the expected complexity of such a cell, regardless of $n$ [18, 19].

In this section we describe an algorithm computing $V(\sigma)$ with a running time dependent on $\tau(r) = \tau_f(r) + \tau_e(r) + \tau_v(r)$, $\tau_f(r)$, $\tau_e(r)$ and $\tau_v(r)$ bounding the expected number of faces, edges and vertices of an additively weighted Voronoi cell amidst $r$ other spheres, respectively. Again, w.l.o.g. we assume that the defining sphere $\sigma$ is centered at the origin.

The algorithm works incrementally by adding the spheres $\sigma_i$ from the input set $S = \{\sigma_i, 1 \leq i \leq n\}$ in their given order. At each step $i$, the algorithm maintains a subdivision $P_i$ of $S^2$ that describes the lower envelope $\min_{1 \leq i \leq n} d_i$ of the functions

$$d_i(p) = \frac{c_i^2 - r_i^2}{2(r_i + \langle p, c_i \rangle)}$$

defined by $\sigma_1, \ldots, \sigma_i$. At certain steps, namely at steps $i = 2^k$, $1 \leq k \leq \lfloor \log n \rfloor$, the algorithm performs a cleanup operation on $P_i$ to guarantee a running time close to $\tau(n)$.

![Figure 2: The edges of an additively weighted Voronoi cell project down as circular arcs.](image)
Additionally, it maintains a set of conflicts \( C_i \). \( C_i \) is a relation between the combinatorial elements of \( P_i \), i.e. the vertices, edges, and faces, and all sites from the set \( \bar{S}_i = S \setminus S^{(i)} \), \( S^{(i)} = \{ \sigma_1, \ldots, \sigma_i \} \). It is used to tell the algorithm, where \( P_{i-1} \) has to be changed when adding \( \sigma_i \). For the analysis, we will turn this algorithm into a randomized one by randomly permuting \( S \) in the beginning.

**The subdivision.** We describe our algorithm restricted to computing \( V(\sigma) \) within the first octant. Eight similar copies will compute \( V(\sigma) \).

For \( 1 \leq i \leq n \) the subdivision \( P_i \) is represented by a collection of polytopes \( H_j(i) \), \( 1 \leq j \leq i \), such that \( \phi_j(H_j(i) \cap S^2) = \{ x \in \partial V : d(x, \sigma) = d(x, \sigma_j) \} \). \( H_0(i) \) represents the unbounded portion of the cell. Set \( \Delta_0 = \{ (x, y, z) \in \mathbb{R}^3 : x \geq 0, y \geq 0, z \geq 0, x + y + z \leq 3 \} \). For \( 1 \leq j \leq i \leq n \) let \( I_j(i) \) be the minimum subset of \( \{0, \ldots, i\} \) identifying non-redundant halfspaces, i.e.

\[
S^2 \cap \Delta_0 \cap \bigcap_{0 \leq k \leq i, k \neq j} h_{j,k} = S^2 \cap \Delta_0 \cap \bigcap_{k \in I_j(i)} h_{j,k}.
\]

Define \( H_i(i) = \Delta_0 \cap \bigcap_{k \in I_j(i)} h_{j,k} \). Recursively we define for \( 0 \leq j < i \)

\[
H_j(i) = \begin{cases} 
\Delta_0 \cap \bigcap_{k \in I_j(i)} h_{j,k} & \text{if } i \text{ is a power of } 2 \\
H_j(i-1) \cap h_{j,i} & \text{if } S^2 \cap H_j(i-1) \cap h_{j,i} \neq S^2 \cap H_j(i-1) \\
H_j(i-1) & \text{otherwise}
\end{cases}
\]

A halfspace \( h_{j,i} \) is redundant in step \( i \) if it does not define an edge of \( P_i \). The cleanup removes redundant halfspaces.

At each step \( i \), the algorithm maintains a canonical triangulation of all \( H_j(i), 0 \leq j \leq i \), for which \( H_j(i) \cap S^2 \neq \emptyset \): 

**Definition 1** Let \( H \subset \mathbb{R}^3 \) be a convex polytope and \( \bar{a} \in \mathbb{R}^3 \) an arbitrary but fixed direction. For each facet \( f \subset H \), choose the minimum vertex \( v_{\text{min}}(f) \) with respect to direction \( \bar{a} \) and triangulate \( f \) towards \( v_{\text{min}}(f) \). The canonical triangulation\(^1\) is obtained as the collection of all 3-dimensional simplices being the convex hull of one of these triangles and \( v_{\text{min}}(H) \), the minimum vertex of \( H \) with respect to \( \bar{a} \).

Let \( S_j(i) \) denote the collection of all those simplices of \( H_j(i) \). We store adjacency information between all simplices \( \Delta \in S_j(i) \), but not between simplices from different sets \( S_j(i) \) and \( S_k(i), j \neq k \). The subdivision \( P_i \) is obtained as the collection of all intersections of these simplices with \( S^2 \). Let \( \Delta \in S_j(i) \) be a simplex obtained by lifting a triangle \( t \) of the triangulation of a facet \( f \) of \( H_j(i) \) towards the minimal vertex \( v_{\text{min}}(H_j(i)) \). Then we set \( v_{\text{min}}(f)(\Delta) = v_{\text{min}}(f) \) and \( v_{\text{min}}(\Delta) = v_{\text{min}}(H_j(i)) \).

\(^1\)One can argue that the present algorithm also introduces an additional prescribed direction. However, contrary to the situation when constructing tangent planes to circular arcs as it occurs in vertical decomposition schemes, ties with respect to \( \bar{a} \) can be broken very easily using, say, a lexicographical ordering of the coordinates, and we do not have to deal with algebraic numbers (or their approximations).
Search structures. On each facet $f \in H_j(i)$, $0 \leq j \leq i$, we maintain a binary tree search structure for point location within $f$. Using red-black trees [12], this structure can be constructed in time linear in the number of vertices $|f|$, queries can be answered in time $O(\log |f|)$, and once the location of insertion is known, the structure can be updated in amortized constant time.

At certain steps, namely if $v_{\min}(H_j(i)) \neq v_{\min}(H_j(i-1))$, we establish a static point location structure for the complete polytope $H_j(i)$ using the following result:

**Theorem 1 ([14])** For any n vertex planar graph, one can build a point location structure of $O(n)$ size in $O(n \log n)$ time, guaranteeing $O(\log n)$ query time.

So, given a polytope $H$ with $n$ facets and a set $M$ of $m$ points, we can determine for each point $p \in M$ the simplex $\Delta$ of the canonical triangulation of $H$ such that $p \in \Delta$, or verify that no such $\Delta$ exists in time $O((n+m) \log n)$.

**Conflict information.** The algorithm maintains three kinds of conflicts that are associated with the vertices, edge fragments and face fragments of all $\Delta \in \bigcup_{j=0}^{i} S_j(i)$:

1. Vertex conflicts: A vertex $v \in P_i$ conflicts with $\sigma_j \in S_i$ if $\phi_j$ will cut the vertex $\phi(v)$ off the cell.

2. Edge conflicts: An edge $e \in P_i$ conflicts with $\sigma_j \in S_i$ if $\phi_j$ intersects $\phi(e)$. We maintain a distinct conflict for each point of intersection. For a single edge $e$ the set of all conflicts $\{(e, \cdot)\}$ is linearly ordered along $e$. This allows us to split an edge $e$ in constant time regardless of the number of conflicts allocated to it.

3. Face conflicts: A face $f \in P_i, \phi(f) \subset \phi_k$ for some $0 \leq k \leq i$ conflicts with $\sigma_j \in S_i$ if $\phi_k \cap \phi_j \neq \emptyset$ and $\phi_k \cap \phi_j \subset \text{int}(\phi(f))$. For each $\Delta$ each face conflict is represented by a point contained in the conflicting region $\Delta \cap S^2$.

We make the following assumptions concerning general position: Any edge conflict is defined as the projection of the intersection of exactly three surfaces $\phi_i, \phi_j, \phi_k$, $0 \leq i < j < k \leq n$, and all points of intersection are not points of tangency.

**Initialization.** The algorithm begins by constructing an unbounded cell represented by $H_0(0) = \Delta_0, S_0(0) = \{\Delta_0\}$ and setting $P_0 = S^2 \cap \Delta_0$. For each $1 \leq i \leq n$ all conflicts of $\sigma_i$ with respect to $P_0$ are calculated. All this can be done in time $O(n \log n)$ and space $O(n)$.

**Update step.** The $i$-th update step when adding sphere $\sigma_i$ to $P_{i-1}$ resulting in $P_i$ is as follows: Let

$$C_{i-1,j}(\sigma_i) = \{\Delta : \Delta \in S_j(i-1), \sigma_i \text{ conflicts } \Delta\}$$
\[ C_{i-1}(\sigma_i) = \bigcup_{j=1}^{i-1} C_{i-1,j}(\sigma_i) \]
\[ A_{i-1}(\sigma_i) = \{ j : 1 \leq j < i, \exists \Delta \in S_j(i-1), \sigma_i \text{ conflicts } \Delta \} \]

For each \( j \in A_{i-1}(\sigma_i) \) we compute the polytope \( H_j(i) \) from \( H_j(i-1) \) using the conflict information \( C_{i-1,j}(\sigma_i) \). We retriangulate the updated part of \( H_j(i) \) and update the search structures on the facets. If no minimal vertex \( v_{\min}(f) \) of a facet \( f \subset H_j(i-1) \) or even the total minimal vertex \( v_{\min}(H_j(i-1)) \) is deleted then this update can be performed within \( O(\vert C_{i-1,j}(\sigma_i) \vert \log(n)) \) time.

For each conflict \( c \) associated with a deleted simplex in \( S_j(i-1) \) we can determine in constant time whether it remains a conflict for a simplex \( \Delta \in S_j(i) \), and, if so, reinsert it as conflict in the updated structure in time \( O(\log(n)) \). Conflicts which are located on a facet that is created for site \( \sigma_i \) are collected into a set \( C_{\text{new}} \).

If we delete a minimum vertex \( v_{\min}(f) \) of a facet \( f \subset H_j(i-1) \), then we retriangulate \( f \) and reallocate all affected conflicts to their new locations. This can be done in time \( O(\vert f \vert + m \log(n)) \), \( m \) being the number of conflicts to be reallocated, and requires no additional space.

Similarly, if the vertex \( v_{\min}(H_j(i-1)) \) happens to be deleted, then we triangulate \( H_j(i) \) from scratch. We compute the point location structure described above and use it to assign all conflicts associated with any \( \Delta \in S_j(i-1) \) to their new locations. This amounts to \( O(\vert H_j(i) \vert \log(n)) \) time and temporarily requires space \( O(\vert H_j(i) \vert) \).

If \( C_{i-1}(\sigma_i) \neq \emptyset \), we compute the polytope \( H_i(i) \), its canonical triangulation, the subdivision \( P_i \), and the point location structures. Note, that exactly sites of simplices in \( C_{i-1}(\sigma_i) \) have facet defining halfspaces \( h_{i,j} \) for \( H_i(i) \). This amounts to time requirements \( O(\vert C_{i-1}(\sigma_i) \vert \log(n)) \) and \( O(\vert C_{i-1}(\sigma_i) \vert) \) space. Then for each conflict \( c \in C_{\text{new}} \), we find \( \Delta \in S_i(i) \) such that \( c \in \Delta \) in \( O(n) \) time, and check if its corresponding site \( \sigma_k, k > i \), associated with \( c \) still conflicts \( \Delta \). If so, we allocate this conflict to \( \Delta \) and recursively visit all neighboring simplices, as long as they have not been visited yet and provided they also conflict with \( \sigma_k \). This traversal can be charged onto the number of newly created conflicts times a factor of \( O(\log(n)) \) for the sorted insertion.

Finally, if \( i \in \{2^k, 1 \leq k \leq \lfloor \log(n) \rfloor \} \), we perform a cleanup operation. For each polytope \( H_j(i) \), such that \( H_j(i) \cap S^2 \neq \emptyset \), we determine the set \( I_j(i) \). We compute the \( H_j(i), 0 \leq j \leq i \), their triangulations, the point location structures, and we reallocate all conflicts to their new locations.

**Probabilistic analysis.** We want to analyze the expected work performed by the algorithm if the \( \sigma_i \) are inserted in random order. We do this, as it is common practice, in the manner described in [7] and [25].

For \( R = \{\sigma_1, \ldots, \sigma_i\} \subset S \) let \( f(R) = \sum_{j=1}^{\ell} \vert S_{ij}(r) \vert \) be the total complexity of the representation of the cell defined by \( R \). Let

\[ f_r = \frac{1}{\binom{n}{r}} \sum_{R \subset S, |R| = r} f(R) \]
be the expectation of this value. In a first step, we will analyze the behavior of our algorithm in terms of $n$ and $f_r$. Then we will bound $f_r$ in terms of $\tau(r)$.

**Proposition 1** Let $S = \{\sigma_i, 1 \leq i \leq n\}$. Then the expected total number of conflicts created or reallocated by the above algorithm when adding the elements from $S$ in random order is bounded by

$$11(n-1)f_1 + \frac{11}{n+1}f_{n+1} + 110n \sum_{r=1}^{n} \frac{f_{r+1}}{r(r+1)} - 121 \sum_{r=1}^{n} \frac{f_{r+1}}{r+1}. $$

**Proof:** Observe, that every simplex of our subdivision is defined by at most 11 spheres. Setting the parameter $d = 11$ in the analysis as given in [25], we obtain the claimed bound. $\Box$

**Proposition 2** Let $S = \{\sigma_i, 1 \leq i \leq n\}$. Then the expected total number of simplices created due to retriangulation operations by the above algorithm when adding the elements from $S$ in random order is bounded by

$$6 \sum_{r=1}^{n} \frac{f_r}{r}. $$

**Proof:** As in the proof of the previous proposition we apply backwards analysis [25]. Let $\Delta$ be a simplex that is created during step $r$ of the algorithm because either one facet or a complete polytope requires retriangulation. Running the algorithm backwards, this is equivalent to the situation that removing $\sigma_r$ from the cell at step $r$ would delete at least one of the two minimal vertices $v_{\min(f)}(\Delta)$ or $v_{\min}(\Delta)$. Each of these vertices is defined by at most 3 sites from $S$. Therefore, the expected value $T_r$ of the number of simplices created due to retriangulation in step $r$ is bounded by

$$T_r \leq \frac{1}{\binom{n}{r}} \sum_{R \subseteq S, |R| = r} \frac{6}{r} f(R) = \frac{6f_r}{r}. $$

Summing up for $r = 1 \ldots n$ we obtain the claimed bound. $\Box$

Because the functions $\tau_f(r)$ and $\tau_e(r)$ might not be monotone increasing in $r$, we set $\hat{\tau}_X(r) = \max_{1 \leq i \leq r} \tau_X(i)$ for $X \in \{e, f, v, \}$. $\Box$

**Proposition 3** Let $S = \{\sigma_i, 1 \leq i \leq n\}$. Then at each step $r$ of the algorithm the following bound holds:

$$f_r \leq 2\hat{\tau}_f(r) + 8\hat{\tau}_e(r) $$

**Proof:** Let $f$ be the number of faces and $e$ the number of edges of an additively weighted Voronoi cell $V$. Then, by Euler’s relation, the number of simplices needed to represent $V$ is bounded by $f + 4e$. Therefore, if $r$ is a power of 2, the cleanup operation guarantees $f_r \leq \tau_f(r) + 4\tau_e(r)$. Otherwise, let $r^* = \lfloor \log r \rfloor$ be the index of the latest
cleanup. Any creation of a new polytope is caused by the creation of at least one new face of the cell, and any addition of a halfspace to a polytope is caused by the creation of at least one new edge of the cell. Hence

\[ f_r \leq \tau_f(r^*) + 4\tau_e(r^*) + \sum_{i=r+1}^{r} \frac{\tau_f(i) + 4\tau_e(i)}{i} \]

\[ \leq \tau_f(r^*) + 4\tau_e(r^*) + \sum_{i=r+1}^{r} \frac{\tau_f(r) + 4\tau_e(r)}{r^* + 1} \]

\[ = \tau_f(r^*) + 4\tau_e(r^*) + \frac{r - r^*}{r^* + 1} \left( \hat{\tau}_f(r) + 4\hat{\tau}_e(r) \right) \]

\[ \leq 2\hat{\tau}_f(r) + 8\hat{\tau}_e(r) \]

\[ \square \]

**Theorem 2** Let \( S = \{ \sigma_i, 1 \leq i \leq n \} \). Then the described algorithm computes the additively weighted Voronoi cell of a sphere \( \sigma \) in expected time

\[ O\left( \sum_{r=1}^{n} \frac{n + r}{r^2} \hat{\tau}(r) \log n \right). \]

Observe, that the time requirements are simply the space requirements times a factor of \( \log n \).

**Proof:** According to propositions 1 and 3 the expected total number of conflicts created is \( O\left( n \sum_{r=1}^{n} \frac{\tau(r)}{r^2} \right) \). Similarly, as shown in proposition 2, the expected total number of simplices created due to retriangulation is bounded by \( O\left( \sum_{r=1}^{r} \frac{\tau(r)}{r} \right) \). Hence, the expected total computational effort required by these steps is bounded by

\[ O\left( \sum_{r=1}^{n} \frac{n + r}{r^2} \hat{\tau}(r) \log n \right). \]

A cleanup operation at step \( i, i \) a power of 2, obviously involves a subset of all operations performed by the algorithm up to step \( i \). Therefore, the expected total computational effort required for cleanup is bounded by the sum

\[ O\left( \sum_{k=1}^{\lfloor \log n \rfloor} \sum_{r=1}^{2^k} \frac{n + r}{r^2} \hat{\tau}(r) \log n \right). \]

\[ \square \]

**Corollary 1** The described algorithm computes the additively weighted Voronoi cell of a sphere \( \sigma \) amidst \( n \) other spheres in expected time \( O(n^2 \log n) \).

If \( \tau(n) = O(n) \), the algorithm accomplishes the computation in expected running time \( O(n \log^2 n) \).
4 Practical considerations

We implemented a version of the algorithm which does not maintain the triangulation of the polytopes $H_j(i)$, but rather uses a planar map on the sphere as underlying data structure. This implies that finding an object $p \in P_i$ a conflict $c$ has to be allocated to needs a search time linear in the size of the face. For this reason, we get an additional worst case factor of $n$ into our running time estimates. The geometric primitives used by the implemented algorithm are formulated in terms of oriented circles, much in the same way as recently proposed in [1].

4.1 Further processing

Both visualization and volume computation require a triangulation of the cell. We do this as follows, see also figure 3:

1. First, we split all edges into line segments. The resulting simple polygons are triangulated.

2. Using the standard Lawson-flip ([17] this triangulation is transformed into a constrained Delaunay-triangulation on the sphere.

3. Finally, circumcenters of large or skinny triangles are added to the triangulation (similar to [24]).

4.2 Preprocessing

Biological macromolecules, such as proteins or enzymatic RNA, usually have thousands of atoms. On the other hand, a typical additively weighted Voronoi cell of one of its atoms has on average about ten neighbors, since these molecules tend to be tightly packed. Therefore, it is important to quickly identify a small set of possible neighbors of the atomic sphere whose cell is to be computed. We achieve this identification using

Figure 3: The subdivision of the surface is refined to a Delaunay triangulation, which is further refined by introducing additional points.
Figure 4: A screen-shot of our interactive program exemplifying the use of additively weighted Voronoi cells in molecular biology.

the lifting map from [2]. Typically, we can identify a set of less than 15 atoms if no hydrogens atoms are present, and less than 30 if they are.

4.3 The Implementation

We implemented the algorithm using the C++ programming language. The algorithm including triangulation requires about 9000 lines of code. Currently, the algorithm uses double precision floating point arithmetic to perform its calculations. Moreover, we incorporated our algorithm with all pre- and post-processing steps included into an interactive program for the examination of data sets from the Brookhaven Protein Database (PDB). See figure 4 for a screen-shot.

For a typical input of up to 30 spheres the algorithm runs well below 100ms, about half of the time being spent on the triangulation. Computing the initial neighbor sets for all atoms of a molecule having about 5000 atoms requires less than 10 seconds. Thus, in less than 10 minutes the volumes of all additively weighted Voronoi cells of all atoms can be calculated. This timing data was obtained on a current mid-range PC offering a 266MHz Pentium II processor and 64MB of RAM operating Windows NT
workstation 4.0. Profiling tools (MS-profile, Intel VTune) reveal that about 60% of the overall time of both the triangulation and the computation of the cell is spent on the evaluation of the numerical primitives.

5 Conclusions

In this paper, we provided new insights into the geometry of additively weighted Voronoi cells. We used these insights to derive a new and relatively simple algorithm for their efficient computation. The algorithm was implemented and performs well on problem instances from molecular biology. Up to now, the design of our algorithm with numerical robustness in mind has proved its worth and could be verified by application to a large collection of data sets taken from the PDB.

References


