On Exact Solution of a Point-Location Problem in a System of d-dimensional Hyperbolic Surfaces

M. L. Gavrilova[*] S. Bespamyatnikh[†]

Abstract. We present a new algorithm for reliable point-location problem in a system of d-dimensional hyperbolic surfaces. The problem is perpetual to many applications, including modeling of a molecular system represented as a set of polydisperse spheres through the use of the Euclidean d-dimensional Voronoi diagram. The algorithm is provided for the d-dimensional case and its implementation in a 3-dimensions is discussed.

Keywords: Reliable computation, hyperbolic surfaces, biological modelling, fixed precision floating-point arithmetic.

1 Introduction

The reliable point-location problem is relevant to many areas, including computer simulation, biological modeling, Geographical Information Systems, motion planning and computer graphics. In these applications, the issues concerning robustness and numerical stability of algorithms, as well as the actual running times of their implementations, are crucial [1, 3, 2, 4, 11].

This paper presents a new and reliable algorithm for addressing the point-location problem based on the implicit computation of the distance from the query point to the d-dimensional hyperbolic surface. The method is based on the fast algorithm to compute the sign of algebraic equation in a fixed-precision arithmetic. The set of hyperbolic surfaces nearest to the query point is first determined by applying a fast approximation algorithm based on the partitioning of the d-dimensional space. The algorithm to carry out the computations in exact arithmetic (with any specified precision) is provided. Finally, the implementation of the algorithms and application to molecular modeling is discussed.

The algorithm can be used for exact and efficient point-location in the d-dimensional Euclidean Voronoi diagram of a set of polydisperse spheres. The main advantage of the algorithm is its simplicity: there is no need to resolve to complex methods based on iterative approximation techniques, or involve libraries for exact computations, such as LEDA, PRECISE, CORE or ECLibrary (Exact Computation Library) [7, 5, 8, 9, 11].

[*]Dept of Computer Science, University of Calgary, Calgary, AB, Canada, T2N1N4. marina@cpsc.ucalgary.ca

[†]Dept of Computer Science, University of Texas at Dallas, Box 830688, Richardson, TX 75083, USA. besp@utd.edu

As a result, much more efficient implementation and simpler method for exact point-location in a system of hyperbolic surfaces, than, for instance, the method described in [5], is obtained. The method is implemented on an example of a 3-dimensional Euclidean Voronoi diagram of spheres, representing the molecular system. Preliminary results confirm the algorithm efficiency.

2 The Problem

Given $(d+1)$ spheres in R^d: $P_i = \{\mathbf{p}_i = (x_{i1}, x_{i2}, ..., x_{id}), r_i\}, i = 1..d+1$, where x_{ij} and r_i are represented by floating-point numbers. The Euclidean Voronoi diagram is used to store the topological information about the system of spheres as well as to answer point-location queries.

Definition 1. A generalized Euclidean Voronoi diagram for a set of spheres S in R^d is the set of Voronoi regions $\left\{\mathbf{x} \in R^d \,\middle|\, d(\mathbf{x}, P) \le d(\mathbf{x}, Q), \forall Q \in S - \{P\}\right\}$, where $d(\mathbf{x}, P)$ is the Euclidean distance function between a point $\mathbf{x}$ and a sphere $P \in S$.

The distance between a point $\mathbf{x}$ and a sphere P with center at $\mathbf{p}$ and radius r_p is defined as

$$d(\mathbf{x}, P) = d(\mathbf{x}, \mathbf{p}) - r_p, \tag{1}$$

where

$$d(\mathbf{x}, \mathbf{p}) = \sqrt{\sum_{i=1}^{d} (x_i - p_i)^2}. \tag{2}$$

According to the definition, the generalized Voronoi vertex is obtained as the intersection of d quasi-halfspaces with hyperbolic boundaries (see Fig. 1). The task is to perform a fast and efficient point-location, i.e. to identify the Voronoi region enclosing the point.

3 The Exact Solution to a Point-location Problem

3.1 The Cell Method Based Approximate Solution

Consider the d-dimensional Voronoi diagram in Euclidean metric. The d-dimensional space is partitioned by the axis-parallel hypercubes in R^d. These are generically called cells in the sequel. A sphere is said to reside in a cell if its center belongs to the cell. Each cell contains a list of particles that currently reside in it. The

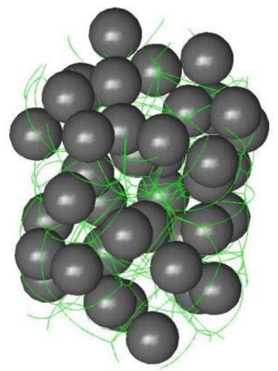

Figure 1: The Euclidean Voronoi diagram of a set of spheres in 3D

set of neighbors of a particle comprises all particles residing in the same or any of the $3^d - 1$ neighboring cells. To ensure correctness, the size of a cell must be greater or equal to the diameter of the largest particle. Then, if two particles are in contact, they are guaranteed to reside in the same or in the two neighboring cells. Each particle $P_i = (\mathbf{p}_i, r_i)$ is represented by a pair consisting of the coordinates of its center $\mathbf{p}_i = \mathbf{p}_i(t)$ and the radius r_i. Assume that the size of the simulation domain is such that there are k cells in each direction. Consider a d-dimensional box with a diameter l as a simulation domain. The size of a cell must exceed the diameter of the largest particle. Thus, k is defined as the diameter of the simulation domain divided by the diameter of a largest particle $M = \max_{P_i \in S}(2r_i)$, i.e. $k = \lceil l/M \rceil$. The diameter of the smallest particle is denoted by $m = \min_{P_i \in S}(2r_i)$.

Assumption 1. The ratio $\gamma = M/m$ between the maximum and the minimum diameter is invariant of n.

Lemma 1 *Under Assumption 1, the maximum number of particles n_c in each cell is bounded by a constant.*

The coordinates of the query point $\xi = (\xi_1, \xi_2, \ldots, \xi_d)$ are checked using direct (non-precise) arithmetic in a constant time and the cell containing the query point is identified. The set of *candidates* - the hyperbolic surfaces intersecting the identified cell - is next tested in order to determine exactly to which region the query point belongs.

The *candidates* selection procedure is now outlined. We will need to establish a couple of basic facts in order to develop the fast method and prove its correctness.

Lemma 2 *Let $\mathbf{A} = \{(a_1, a_2, ..., a_d), r_a\}$ and $\mathbf{B} = \{(b_1, b_2, ..., b_d), r_b\}$ be two d-dimensional spheres. Then for any point $x \in \mathbf{B}$, $d(b, \mathbf{A}) - r_b \le d(x, \mathbf{A}) \le d(b, \mathbf{A}) + r_b$, where $b = (b_1, b_2, ..., b_d)$ is the center of the sphere $\mathbf{B}$.*

Lemma 3 *Let $\mathbf{R}$ be a d-dimensional box with center r and sides r_i, $i = 1..d$. Let $\mathbf{A} = \{(a_1, a_2, ..., a_d), r_a\}$ and $\mathbf{B} = \{(b_1, b_2, ..., b_d), r_b\}$ be two d-dimensional spheres. If the following condition is true: $d(r, \mathbf{A}) + r_R < d(r, \mathbf{B})$, where $r_R = \sqrt{\sum_{i=1}^{d} r_i^2}$ is the radius of the smallest sphere enclosing R, then for any point $x \in \mathbf{R}$, $d(x, \mathbf{A}) < d(x, \mathbf{B})$.*

Following Lemma 3, $\mathbf{R}$ does not contain any points from Voronoi region $Vor(\mathbf{B})$. The above Lemma enables us to limit the total number of spheres that must be considered as potential candidates for Voronoi diagram point location. The following algorithm is used to pre-process the set of spheres.

For each cell $\mathbf{R}$, compute distances from the center of the cell r to each of the spheres. Choose the closest sphere A and exclude all spheres for which the distance $d(r, \mathbf{B})$ is greater than $d(r, \mathbf{A}) + r_R$. Essentially, this allows to compute the set of spheres, whose Voronoi regions might intersect the cell $\mathbf{R}$.

Now, we can subdivide the query space into a set of regular cells, and compute the set of candidate spheres for each of the cells.

If the resulting set of candidate spheres it too large, it is possible to divide $\mathbf{R}$ into two parts and recompute the set of potential candidates. We will continue subdividing the cells until the number of potential candidate spheres is sufficiently small, or if the size of the cell becomes sufficiently small.

Note that for some sphere configurations, it is possible that the number of candidates is very large. One such example is a set of spheres situated on a larger sphere around $\mathbf{R}$. However, on average, the number of candidates is just a constant.

If a regular cell structure was selected, then the cell containing the query point q can be located in constant time. If cells were obtained by subdividing larger cells, then the cell containing the query point can be located in $\log(m)$ time, where m is the total number of cells, by walking the $k-d$ tree of cells.

After the cell containing the query point is found, we compare the distances from the query point to each of the candidate spheres that were precomputed for this cell. Assume the number of candidates is k. Using the exact comparison method described in the following section, we determine the closest sphere in $O(k)$ time.

3.2 The Exact Sign of Expression Algorithm

The ESSA algorithm computes the sign of an algebraic sum exactly using standard fixed precision floating-point arithmetic [6]. The summands are represented as normalized single precision floating-point numbers with fixed mantissa length. The main idea of the algorithm is to separate the positive and the negative summands into two lists, sort the two lists separately and then decimate leading terms recursively. More details on algorithm implementation can be found in [6]. The ESSA method is extended with the algorithm to compute the sign of arithmetic expression containing product exactly. In this approach, a product of k single-precision terms $N_1 * N_2 * ... * N_k$ is represented as a sum of 2^{k-1} single precision monomials. The method is described in [5]. Thus, the ESSA can be applied to compute the sign of algebraic expression containing sums and products exactly.

3.3 The Exact Point Location Algorithm

The task

Given a query point $\xi = (\xi_1, \xi_2, ..., \xi_d)$ and two spheres $\mathbf{a} = \{(a_1, a_2, ..., a_d), r_a\}$ and $\mathbf{b} = \{(b_1, b_2, ..., b_d), r_b\}$. It is required to determine whether the query point ξ is closer to the sphere **a** or **b**.

The Solution

Lemma 4 *In exact computation of the sign of expression $x-y$, where both x and y are non-negative, the sign of $x-y$ is the same as the sign of expression x^2-y^2 computed exactly.*

Assume without the loss of generality that $r_a - r_b \geq 0$. Then comparing the values of d_a and d_b exactly will provide an exact answer to the problem:

$$d_a = \sqrt{\sum_{i=1}^{d} (\xi_i - a_i)^2} - (r_a - r_b)$$
$$d_b = \sqrt{\sum_{i=1}^{d} (\xi_i - b_i)^2}$$

If $d_a < d_b$ then the query point ξ is closer to sphere **a**, if $d_a = d_b$ then the query point ξ is on the bisector, and if $d_a > d_b$ the ξ is closer to sphere. The following present the exact algorithm to determine whether the query point is closer to the sphere **a** or **b**.

The Algorithm

1. Compute the sign of d_a exactly using the ESSA. Note that d_b is always non-negative. To compute the sign on d_a, consider the two parts of the equation through which it is expressed. Both summands are positive. Take both parts to the power of 2 and compute the sign of their difference exactly using ESSA

$$\delta = \sum_{i=1}^{d} (\xi_i - a_i)^2 - (r_a - r_b)^2 .$$

If the sign δ is negative, then d_a is negative. Report that the query point ξ is closer to sphere **a**. Otherwise, goto Step 2.

2. Compute the sign of the difference of d_a and d_b, taking to the power of 2.

Since both d_a and d_b are positive, take them to the power of 2, and compute the sign of the difference of squares exactly:

$$d_a^2 - d_b^2 = \sum_{i=1}^{d} (\xi_i - a_i)^2 + (r_a - r_b)^2$$
$$-2(r_a - r_b)\sqrt{\sum_{i=1}^{d} (\xi_i - a_i)^2} - \sum_{i=1}^{d} (\xi_i - b_i)^2 \quad (3)$$

In order to do it, first compute the sign of the following expression exactly:

$$\sum_{i=1}^{d} (\xi_i - a_i)^2 + (r_a - r_b)^2 - \sum_{i=1}^{d} (\xi_i - b_i)^2 \quad (4)$$

Apply the ESSA to handle this task, using the technique to take the equation to the 2nd power described in [5].

If the sign of (4) is negative, then the conclusion can be drawn that the sign of (3) is also negative, since

$$2(r_a - r_b)\sqrt{\sum_{i=1}^{d} (\xi_i - a_i)^2} \quad (5)$$

is always positive. Report that the query point ξ is closer to sphere **a**. Otherwise, goto Step 3.

3. Since both terms of (3) are positive, taking them to the second degree will lead to the equation:

$$\left(\sum_{i=1}^{d} (\xi_i - a_i)^2 + (r_a - r_b)^2 - \sum_{i=1}^{d} (\xi_i - b_i)^2\right)^2$$
$$-4(r_a - r_b)^2 \sum_{i=1}^{d} (\xi_i - a_i)^2 \quad (6)$$

Compute the sign of this expression exactly using the ESSA. Note that most of the summands are the 4th degree products. If the sign of equation (6) is negative, then the query point ξ is closer to sphere **a**. Else if the sign of equation (6) is 0 then the query point ξ is located on the hyperboloid (bisector between two spheres). Otherwise, the the query point ξ is closer to sphere **b**.

Lemma 5 *The above algorithm determines location of a query point in respect to the Voronoi face, represented as a hyperboloid in a d-dimensional space, exactly in a floating-point arithmetic.*

4 Experimental results

The method feasibility was verified on an example of a 3D Euclidean Voronoi diagram of a set of spheres, representing molecular system. Algorithms were implemented in object-oriented C++ environment on 1.4 MHz Intel processor. All operations on floating-point numbers were performed using the ESSA algorithm and its modification, the ESAE (Exact Sign of Algebraic Expression) method (see [7]).

The experiments were conducted on two configurations: one representing a random distribution of spheres confined inside a cube, another is a close to a degenerate case when four spheres in 3D are almost tangent to the same 3D planes. Note that in the second case the coordinates of the spheres are slightly disturbed in order to make it possible to find a solution (i.e. the inscribed sphere).

First, the number of iterations required to obtain the exact to a specified number of bits solution was measured. The computations were carried out to compute the 53 bits representing a mantissa of a floating-point number exactly. Experiments showed that on average only two iterations were required to obtain the desired precision.

Next, we measure the precision of computation for random and degenerate distributions. The precision depends on the number of iterations and is increased only twice for both configurations. The method outperforms the approach based on the exact computation of distances to two orders of magnitude.

5 Conclusions and Discussion

The new algorithm for fast and precise point-location problem for d-dimensional Euclidean Voronoi diagram of a set of spheres is provided. Results obtained confirm the algorithm correctness and demonstrate suitability of the method for reliable computation of the Euclidean Voronoi diagram representing molecular system. The method possesses a number of important characteristics, such as simplicity, easy of implementation and possibility of bringing to different platforms. The algorithm also allows for virtually unlimited precision exact arithmetic and is more efficient than the previously developed method using ECLibrary or LEDA implementation. We plan to use the designed algorithm in applications in computational biology where the point-location queries are required to compute the nearest atom.

The first author would like to thank the U of C Research Services and NSERC for the valuable support of the project.

References

[1] Blum, L., Cucker, F., Shub, M. and Smale, S. "Complexity and Real Computation" (1997).

[2] Dey, T.K., Sugihara K. and Bajaj, C. L. DT in three dimensions with finite precision arithmetic, Comp. Aid. Geom. Des **9**(1992) 457-470.

[3] Edelsbrunner, H. And Mcke, E. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, SoCG (1988) 118-133.

[4] Fortune, S. And Wyk, C. Efficient exact arithmetic for computational geometry, SoCG, (1993) 163-172.

[5] Gavrilova, M. A Reliable Algorithm for Computing the Generalized Voronoi Diagram for a Set of Spheres in the Euclidean d-dimensional Space, CCCG (2002) 82-87.

[6] Gavrilova, M., Ratschek, H. and Rokne, J. Exact computation of Voronoi diagram and Delaunay triangulation, Reliable Computing, (2000) 6(1) 39-60.

[7] Gavrilova, M. (2002) Algorithm library development for complex biological and mechanical systems. DIMACS Workshop on Implementation of Geometric Algorithms, 2002.

[8] S. Krishnan, M. Foskey, T. Culver, J. Keyser, D. Manocha, PRECISE: Efficient Multiprecision Evaluation of Algebraic Roots and Predicates for Reliable Geometric Computations, SoCG (2002).

[9] Naher, S. The LEDA user manual, Version 3.1 (Jan. 16, 1995). Available from ftp.mpi-sb.mpg.de

[10] Sugihara, K. And Iri, M. A robust topology-oriented incremental algorithm for Voronoi diagrams, IJCGA, (1994) 4 (2): 179-228.

[11] Yap, C., Dube, T. The exact computation paradigm, In "Computing in Euclidean Geometry" (2nd Edition). Eds. D.-Z. Du and F.K. Hwang, World Scientific Press (1995).