Degree-driven algorithm design for computing the Voronoi diagram

David L. Millman * Jack Snoeyink †

Department of Computer Science, University of North Carolina at Chapel Hill

1 Introduction

The Voronoi diagram is the classic proximity query structure. It finds the nearest point from a finite set S to a given query point q and it is the partition of the plane into the maximal connected regions. The Voronoi diagram's topological structure can be computed using four times the precision of the input, but representing its vertices requires five times the input precision. Furthermore, using the Voronoi diagram to solve proximity queries in $O(\log n)$ requires six times the precision of the input and query points. Liotta, Preparata, and Tamassia [3] have derived a structure from the Voronoi diagram whose computation requires five times the precision of the input, but which supports proximity queries in $O(\log n)$ time, using only two times the input precision. In this paper, we show how this structure can be computed directly, using at most triple precision in $O(n(\log n + \log g))$ time where g is the bisector length.

Computing Voronoi diagrams is a well studied problem and many optimal algorithms have been proposed [1]. These algorithms rely on correct predicates to handle the numerical issues that arise upon implementation but do not let the complexity of predicate implantation affect the design of the algorithm.

Methods for handling numeric issues in geometric algorithms fall into a few categories: rounding, exact geometric computation [5], arithmetic filters [2], topological consistency [4] and degree-driven algorithmic design [3]. This paper falls into the last group of degree-driven design.

2 The Cell Graph

Given a set of n sites $S = \{s_1, s_2, \ldots, s_n\}$ whose coordinates are b-bit integers, we would like to construct the *implicit Voronoi diagram* $V^*(S)$ of [3], which consists of a topological component, the planar embedding represented by a suitable structure such as a doubly connected edge list [1] and a geometric component, which for each vertex (v_x, v_y) of the Voronoi diagram of S, the implicit Voronoi diagram, $V^*(S)$ stores the half integers v_x^*, v_y^*, where $v_x^* = v_x$ when v_x is an integer and $v_x^* = \lfloor v_x \rfloor + \frac{1}{2}$ when v_x is not. We define v_y in a similar manner.

To construct $V^*(S)$ we create a structure called a *cell graph* that encodes all the information of $V^*(S)$ and maintains connectivity and orientation information used for incremental updating. Any Voronoi vertex on a grid point and any grid cell containing one or more Voronoi vertices is a *cell vertex*. When differentiation is necessary, we refer to the former as a *grid cell vertex* and the latter as a *non-grid cell vertex*. For each edge $(u, v) \in V(S)$ if u is mapped to c_u and v is mapped to c_v, then the edge $(c_u, c_v) \in C(S)$. We call (c_u, c_v) a *cell edge*. Each cell edge corresponds to a Voronoi edge, b, and stores the two sites defining b. In addition, each cell vertex v maintains a circular doubly-linked list of cell edges with the same ordering as Voronoi edges entering the grid cell that v represents.

From the cell graph we can create the structure of [3] by leaving grid cell vertices where they are and snapping non-grid cell vertices to half gird points. In the following sections we describe the predicates and operations used in a randomized incremental construction for the cell graph.

3 Predicates and Operations

Two sites s_1, s_2 on a grid determine a bisector, b_{12} that partitions the grid into points closest to s_1, s_2 or equidistant and on b_{12}. Using this observation we describe predicates for testing a grid cell against a bisector, enumerating grid cells containing bisectors and locating grid cells of bisector intersections.

Given two sites s_1, s_2 and a grid cell G, deciding

*dave@cs.unc.edu

†snoeyink@cs.unc.edu

if the bisector of s_1 and s_2 is in G takes constant time and requires degree two computation, through computing the squared distances of the grid points of G. Furthermore we can use the information derived in this predicate to decide the cardinal directions that a bisector stabs a grid cell if at all. We call this the *bisectorInCell* predicate.

Given two sites s_1, s_2 and a grid cell G that b_{12} stabs we can compute the intersection points of b_{12} and the grid cell using degree three computations in constant time. Usually, the intersection of two bisectors requires degree five predicates, but by taking advantage of the fact that we are intersecting b_{12} with horizontal or a vertical line, we can simplify the computation.

Given s_1, s_2, and a direction to walk, we define a *bisectorWalk* operation to be a traversal of a subset of the cells that b_{12} passes though. Using the bisectorInCell predicate, we can walk b_{12} starting at the midpoint of $\overline{s_1 s_2}$ with computations of maximal degree two in $O(\log g)$ where g is the length of the walk.

The previous predicates and operations can be used to compute the *bisectorIntersection* operation. Given four non collinear sites $\{s_i, i = 1, \ldots, 4\}$, the grid cell that contains the intersection of the bisector of s_1, s_2 and s_3, s_4 can be computed using maximal degree three predicates in $O(\log g)$, where g is the distance between the intersection of the bisectors and the midpoint of segment $\overline{s_1 s_2}$.

4 Incremental construction

Next we describe the randomized incremental construction (RIC) [1] of the cell graph, and begin with three non-collinear sites s_1, s_2, s_3. As these sites are non-collinear they form a Voronoi vertex. Using the bisectorIntersection operation with b_{12} and b_{13}, we can determine the grid cell where this vertex resides.

Assume that we have already constructed a cell graph of $n-1$ sites and that we would like to insert site s_n. We proceed in the standard RIC fashion for computing Voronoi diagrams, by first locating the cell containing s_n and carving out the region of s_n similar to the method used in [4]. Instead of the normal RIC method that actually computes the intersection point of bisectors we use the bisectorIntersection operation to find the grid cell G where the intersection of two bisectors occur.

Finding G has two cases, either there is no cell vertex corresponding to G or one already exists. In the first case, we just create a new cell vertex and update cell edges. Care must be taken if a cell vertex c_v already exists. First, we add the new cell edge to c_v appropriately ordered. Next we walk around the cell edges of c_v in the direction that cell edges are removed. This can be determined by the side of the bisector s_n is on.

As each insertion will cause an expected constant number of vertices, faces and edges to be modified the only additional penalty we incur with this method as apposed to the standard RIC algorithm is the $O(\log g)$ for each segment intersection. This gives us that the cell graph of n sites can be constructed in $O(n(\log n + \log g))$ where g is the bisector length using computations of maximal degree three. As the the cell graph subsumes all the information encoded in $V^*(S)$, it can also support nearest neighbor queries in $O(\log n)$ time with degree two computations.

5 Conclusions

This paper presented a method for computing the implicit Voronoi diagrams of [3] in $O(n(\log n + \log g))$ where g is the bisector length using a maximal of degree three predicates. Furthermore, to our knowledge this is the first algorithm that allows for such a construction without fully computing the Voronoi diagram.

References

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Santa Clara, CA, USA, 2008.

[2] O. Devillers and F. P. Preparata. Further results on arithmetic filters for geometric predicates. *Comput. Geom. Theory Appl.*, 13(2):141–148, 1999.

[3] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries: an illustration of degree-driven algorithm design. In *SCG '97*, pages 156–165, New York, NY, USA, 1997. ACM.

[4] K. Sugihara and M. Iri. Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic. *Proceedings of the IEEE*, 80(9):1471–1484, 1992.

[5] C.-K. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7(1-2):3–23, 1997.