1 Introduction

The Voronoi Diagram (VD, for short) is a ubiquitious structure that appears in a variety of disciplines - biology, geography, ecology, crystallography, to mention just a few.

The VD and its dual, the Delaunay Triangulation (DT, fot short), are two of the most important and fundamental data structures in Computational Geometry that have been succesfully applied to a variety of proximity problems. We will study a few of these later. It is not surprising therefore that these have received a lot of attention from researchers, leading to a very extensive literarure on their properties and applications.

In this chapter we will first describe what is a VD and discuss some of its basic properties. Next we will discuss algorithms for computing the VD and explore some of its applications in solving other Computational Geometry problems.

We will then introduce the DT, discuss some of its fundamental properties and show how to compute a DT directly from a given set of points.

2 Voronoi diagrams

Three ingredients are needed to define a VD: a set of objects, an ambient space in which these objects are embedded and a notion of distance between a point of the ambient space and a subset of the given set of objects. The distance function enables us to define bisectors of pairs of sets of objects that partition the ambient space into a VD of the given set of objects. Here's a concrete and simple example.

Let the set of objects be a set S of n sites (points) $\{s_1, s_2, \ldots, s_n\}$, the ambient space the two dimensional plane and the distance be the Euclidean distance between any point of the plane and a site in S.

The bisectors between pairs of sites are straight lines that partition the plane into n convex regions, one corresponding to each site s_i. The region of site s_i is called the *Voronoi polygon* of s_i and consists of the points which are *closer* to site s_i than to the remaining sites in S. The resulting partition of the plane is called a (nearest point) VD of S.

Figure 1 below shows a Voronoi diagram on 4 sites.

The vertices and edges that make up the boundaries of the convex regions are called *Voronoi edges* and *Voronoi vertices* respectively.

As is obvious, by varying the three ingredients we mentioned above we get a wide variety of Voronoi diagrams (see [2] for the amazing variety of possibilities). While in this chapter we will be mainly

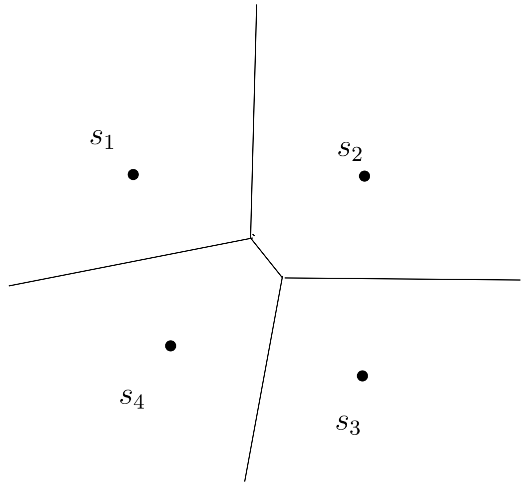

Figure 1: *Voronoi diagram of 4 points*

concerned with Voronoi diagrams of a set of point sites, we will mention a few of the more well-studied ones at the end of the chapter.

3 Properties of a nearest point Voronoi Diagram

We establish a few useful properties of this diagram, assuming that no four points are co-circular (that is, lie on the same circle).

Claim 1 *The number of Voronoi vertices and edges are respectively* $2(n-1)-h$ *and* $3(n-1)-h$ *respectively, where* n *is the number of sites and* h *the number of sites on the convex hull of* S.

A finite graph satsfies Euler's formula $V - E + F = 2$ for a convex polyhedron, where V, E and F are respectively the number of vertices, edges and faces of the polyhedron (graph). This can be established by a stereographic projection of the polyhedron, embedded on the surface of a sphere, into a finite graph.

The Voronoi diagram of a set of n sites can be transformed into a finite graph by enclosing the Voronoi diagram inside a very large circle and treating each of the h (= # of convex hull vertices of the given point set) arcs that lie inside an unbounded Voronoi polygon to correspond to an edge of the graph.

For this finite graph, we use the fact that $2E = 3V$ (each side of the equality counts the total degree of the graph) to deduce, using Euler's formula, that $V = 2(F-2)$ and $E = 3(F-2)$.

Now $V = h + \#\ Voronoi\ vertices$ and $E = h + \#\ Voronoi\ edges$. Hence the number of Voronoi vertices $= 2(F-2) - h$ and the number of Voronoi edges $= 3(F-2) - h$. Since $F = n+1$, these counts are therefore $2(n-1) - h$ and $3(n-1) - h$ respectively $\square$.

Claim 2 *Every Voronoi vertex is of degree 3.*

Proof: Otherwise, 4 or more points would be co-circular, contrary to our assumption.

The next property is significant.

Claim 3 *The circle $C(v)$, centered at a Voronoi vertex v, and passing through the sites that define v has no other sites in its interior.*

Proof: By the definition of v it is closest to the sites that define it; a site in the interior of $C(v)$ by being closer to v would contradict this definition. (see Fig 2).

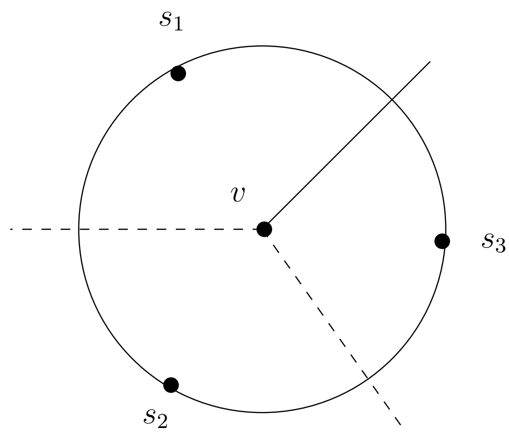

Figure 2: No sites in the interior of $C(v)$

Another interesting question is this: which bisectors define the edges of the Voronoi polygon of a site s_i ?

Claim 4 *The bisector of the sites s_i and s_j defines an edge of the Voronoi polygon of s_i (and thus symmetrically that of the Voronoi polygon of s_j) iff there exists a point of the plane whose nearest sites are both s_i and s_j simultaneously.*

Proof: (only if) Let the site s_j contribute an edge to the Voronoi polygon of s_i. Let p be an internal point on this edge. If s_i and s_j are not the nearest sites of p, let s_k be a site that is closer. From Fig. 3 it is clear that this redefines the edge that s_j contributes to the Voronoi polygon of s_i to exclude p. This contradicts the assumption that p is an internal point of the edge of the Voronoi polygon of s_i due to s_j. Thus p cannot have a site closer to it than s_i and s_j.

(if) Let there exist a point p whose nearest sites are both s_i and s_j. Thus p must lie on the bisector of s_i and s_j. If we move p slightly along $\overline{ps_j}$ towards s_j, then s_j becomes the closest site of p, putting it in the Voronoi polygon of s_j; in the same way, by moving it along $\overline{ps_i}$ towards s_i, we put it in the Voronoi polygon of s_i. Thus p must belong to the part of the bisector of s_i and s_j that is a common edge of their Voronoi polygons. $\square$

The next claim shows a nice connection beween the VD of a set of points and its convex hull.

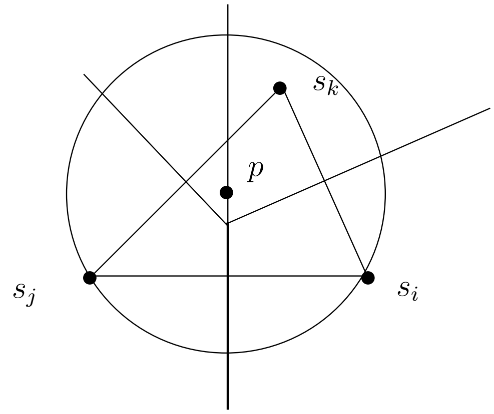

Figure 3: *When p is closer to s_k than s_i or s_j ....*

Claim 5 *The Voronoi polygon of s_i is unbounded iff s_i is a point on the boundary of the convex hull of S.*

Proof: (only if) Let s_i be a site interior to the convex hull of S. This implies that s_i lies inside the triangle formed by some triplet of sites on the hull boundary. The bounded region formed by the bisectors of s_i and each of these three sites contains the Voronoi polygon of s_i. Hence the Voronoi polygon of s_i is bounded.

(if) Let s_i be a site on the hull boundary of S. Let s_j and s_k be sites adjacent to it on the hull boundary. Consider a supporting line l of the convex hull through s_i. Let p be a point on a ray through s_i that is orthogonal to l and $C(p)$ be a circle centered at p with radius $|ps_i|$ (see Fig. 4). The point p is in the Voronoi polygon of s_i, since by construction any other site is farther from it than the radius of $C(p)$.

Since p is an arbitrary point on the line orthogonal to l, the Voronoi polygon of s_i is unbounded. $\square$

The dual of the Voronoi diagram is obtained by joining pairs of sites whose Voronoi polygons are adjacent. We next show that:

Claim 6 *The dual of the Voronoi diagram of S is a triangulation* [1] *of S.*

Proof: We first show that no two segments in the dual diagram intersect except at their end points.

Assume that the segments joining the sites s_i and s_j intersect the segment joining the sites s_u and s_v at the point p.

If p is the mid-point of both segments then p would have to belong to the Voronoi polygons of the sites s_i, s_j, s_u and s_v. This is impossible as a point can belong to at most three different Voronoi

[1] Given n points in the plane, join them by nonintersecting straight line segments so that every region internal to the convex hull is a triangle.

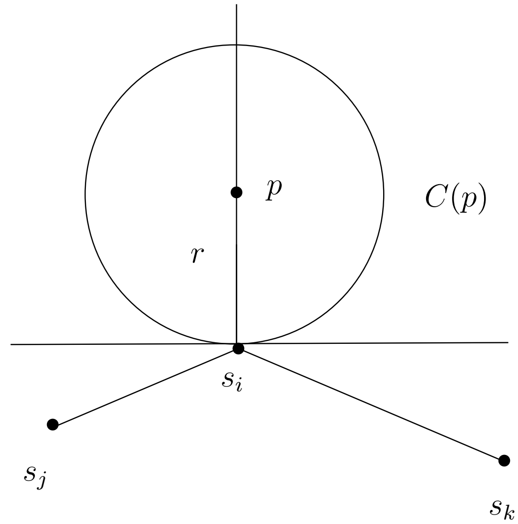

Figure 4: *A convex hull vertex of S has an unbounded Voronoi polygon*

polygons.

Thus p lies on one side of the mid-point of at least one of the two segments, say $\overline{s_i s_j}$.

If p is the mid-point of the other segement, then p is on the boundary of the Voronoi polygons of s_u and s_v, and in the interior of the Voronoi diagram of s_i as well. This is not possible.

Otherwise p is closer to, say s_u, than s_v. Hence p is in the interior of the Voronoi polygons of s_i as well as s_v (see Fig. 5). This is also impossible.

Thus in all cases we conclude that the segments $\overline{s_i s_j}$ and $\overline{s_u s_v}$ cannot cross.

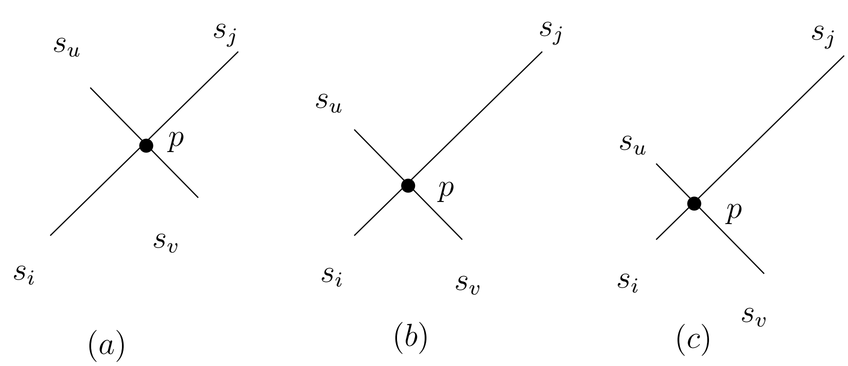

Figure 5: *Non-intersection of two segments in the dual diagram*

Next, we show that the set of segments in the dual diagram is maximal.

Suppose otherwise. This implies we can add a segment joining two sites s_i and s_j whose Voronoi diagrams are non-adjacent, while still satisfying the non-intersection property.

Thus the set of edges joining pairs of sites due to the dualization form a maximal set. □

This triangulation is the famous Delaunay triangulation on S.

4 Constructing a Voronoi diagram

Many algorithms are available for constructing a (nearest point) Voronoi diagram. An algorithm based on the divide-and-conquer paradigm is described in the textbook by Preparata and Shamos [1]. This algorithm is messy and hard to understand.

Here we will describe a simple and beautiful algorithm by Fortune [] that is based on the sweepline paradigm. The surprising aspect of this algorithm is the applicability of the sweepline technique to this problem - something that appears impossible at first sight.

Here, we will indulge in a brief digression to explain the sweepline technique so that the cleverness of Fortune's algorithm is better appreciated.

The sweepline technique as the very name suggests is a method by which we sweep a line through a set of geometric objects from left to right (or right to left, if you wish) in order to compute some function on these objects. For example, given a set of line segments in the plane (again!) we might be interested in determining all pairs of segments that intersect or, maybe, just obtain a count of the number of interesecting pairs.

This can be accomplished with two simple data structures - a sweepline structure that keeps track of all the line segments that intersect the current position of the sweepline and an event queue that keeps track of the discrete set of events where the sweepline structure changes (for more details see Preparata and Shamos [1]).

The significant fact for us to notice is that an individual segment appears in the sweepline structure at the event corresponding to its left end-point and disappears from the sweepline at the event corresponding to its right end-point. All the intersections due to this segment are discovered beween these two events, so that once this segment leaves the sweepline status no new intersections due to this segment remain to be discovered.

If we try to apply this technique in a straightforward way to computing the Voronoi digram of a set of points we immediately encounter the problem that the Voronoi polygon of a point (this is an event) extends on both sides of this event. In other words, the computation of the Voronoi polygon of a point is not complete when the sweepline goes past the point.

Fortune proposed a novel way of getting around this difficulty. In addition to the usual sweepline infrastructure, the effect of a point-event is maintained in the form of a parabola that is generated when the sweepline reaches a point. This parabola has the point as its focus and the sweepline as its directrix so that it trails the sweepline. and remains active till the Voronoi polygon of the point that caused its generation has been constructed.

All active parabolas are maintained as a beach-front made up of parabolic arcs (in mathematical terms, this is is an upper envelope of all the active parabolas) that changes dynamically as new parabolas are added and inactive parabolas are removed.

An intriguing question that crops up is how does a parabola best represent the interest of an event point. To answer this question we note that the Voronoi edges are generated by the intersection of a pair of parabolas that are adjacent on the beach line. A Voronoi vertex is generated when three parabolas on the beach-front meet at a point. This point is equidistant from the points that are the foci of these three parabolas as well as from the sweepline which is the directrix of all three. The circle centered at this common intersection point, with radius equal to distance from the directrix is tangent to the directrix, passing through the foci af all the three parabolas, and is thus called a tangent-event. At this time, a Voronoi vertex is generated and the "middle" of the three parabolas disappears from the beach-front.

5 Delaunay Triangulation

As explained earlier, the dual of the Voronoi diagram of S is a (the ?) Delaunay Triangulation (DT, for short from now on) on S, and as such it is a byproduct of the Voronoi diagram construction algorithm.

However, it is possible to construct a DT directly from the given set of points S. Below, we discuss such an algorithm based on the incremental construction paradigm.

A Delaunay triangulation on S is (uniquely?) characterized by the property that the circumcircle of each triangle contains no other sites in its interior. This folllows from the empty circle property of a Voronoi diagram described in the previous section.

In the algorithm below, we will need the following characterization of an edge in the DT.

Claim 7 *An edge connecting two sites s_i and s_j is an edge in the DT iff there exists a circle passing through s_i and s_j that does not contain any site in its interior.*

5.1 Incremental Algorithm

Assume that we have constructed the DT of the first $i-1$ sites ($i > 3$), DT_{i-1}. To update this triangulation upon addition of the point p_i, we first locate the triangle $\triangle rst$ of DT_{i-1} in which p_i lies. Then the edges $\overline{p_i r}$, $\overline{p_i s}$ and $\overline{p_i s}$ are new Delaunay triangulation edges (Claim 7).

We show, for example, that $\overline{p_i r}$ is a Delaunay edge. Let $\overline{rr'}$ be a diameter of the circle circumscribing triangle $\triangle rst$ (see Fig. 6). Choose r'' on this diameter so that $p_i r''$ and $p_i r$ are orthogonal. The circle on rr'' as diameter goes through the edge $\overline{p_i r}$, and has no site in its interior because it is entirely inside the circumcircle of $\triangle rst$.

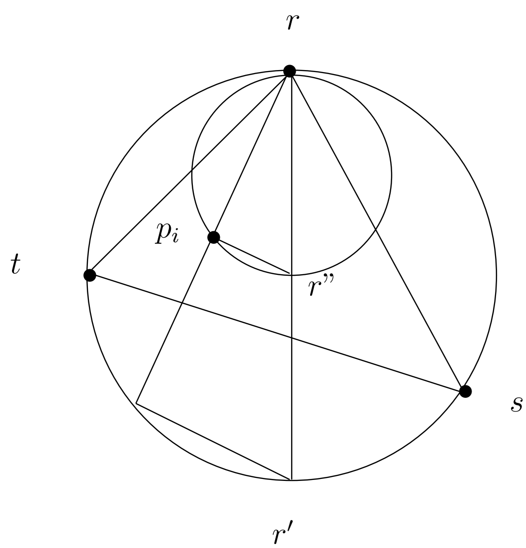

Figure 6: New Delaunay edges incident on p_i

A triangle $\triangle T$ in the current triangulation is said to be in *conflict* with p_i if the circumcircle of triangle $\triangle T$ contains p_i and thus cannot be part of DT_i. Thus the status of the edges $\overline{rs}$, $\overline{st}$ and $\overline{tr}$ need to be checked. Consider the edge $\overline{rs}$, and the triangle $\triangle prs$, not containing p_i. If triangle $\triangle prs$ is in conflict with p_i then the edge $\overline{rs}$ is not a Delaunay edge and we replace it by the new edge $\overline{pp_i}$. This is called edge-flipping. Every time we do an edge-flipping two new edges are up for the circumcircle test. We continue till no edge-flipping occurs, when we have the Delaunay triangulation of the i sites.

5.1.1 Analysis of the Incremental Algorithm

The following gross analysis tells us that the algorithm is in $O(n^2)$. The number of edge-flippings caused by the insertion of p_i is proportional to the degree of this vertex in the triangulation. The degree of each vertex is in $O(i)$ and hence the total number of edge-flippings after the insertion of the n-th point is in $O(n^2)$. The cost of a brute-force location of the point p_i in DT_{i-1} is also in $O(i)$ and hence the cost over the entire sequence of n insertions is in $O(n^2)$.

A more subtle (and messy!) analysis shows that the *expected complexity* of this algorithm is in $O(n \log n)$.

The first observation is that the average degree of a site in the DT is at most 6, since there are at most $3n-6$ edges. Thus if we make a random sequence of insertions the expected number of edge-flippings is in $O(n)$.

5.2 The Delaunay Tree Data Structure

The *Delaunay Tree* data structure provides a more efficient alternative to the brute-force search for finding a triangle in DT_{i-1} that contains p_i. It is a layered *DAG* (short for Directed Acyclic

Graph) in which we maintain all triangles created during the incremental construction. The i-th layer consists of traingles created during insertion of site p_i.

The root (0-th layer) of this *Delaunay tree* is a large triangle that contains all the sites of S. From each each node (storing a triangle) at a given layer, we maintain pointers to all nodes in the next layer that store triangles that overlap with this triangle. Referring to the discussion above, we thus keep pointers from the nodes that stores the old triangles $\triangle rst$ and $\triangle prs$ to the triangle $\triangle pp_is$ if it is in DT_i.

To locate p_i in the DAG corresponding to DT_{i-1}, we start at the root of the DAG and follow the pointers to descend along a path in which the triangles are in conflict with p_i till we hit a leaf node which stores the triangle that contains p_i.

We first prove the following claims.

Claim 8 *If the triangle of DT_{i-1} containing p_i is known, the structural work needed for computing DT_i from DT_{i-1} is proportional to the degree of p_i in DT_i.*

Claim 9 *For each $h < i$, let d_h denote the expected number of triangles in $DT_h \setminus DT_{h-1}$ that are in conflict with p_i, then $\sum_{h=1}^{i} d_h = O(\log i)$*

Proof: Let C denote the set of triangles in DT_h that are in conflict with p_i. A triangle $T \in C$ belongs to $DT_h \setminus DT_{h-1}$ iff it has p_h as a vertex. Since the number of triangles in DT_h is $2*h-5$ and the expected degree of p_h is 6 the probability that a triangle in conflict with p_i survives is $6/(2*h-5) = 3/h$, under the assumption that p_h is randomly chosen. Thus, the expected number of triangles in $C \setminus DT_{h-1}$ is $3*|(C)|/h$. Since the expected size of C is less than 6(because the average degree of p_i is at most 6), therefore $d_h < 18/h$. Thus , $\sum_{h=1}^{i} d_h = O(\log i)$. □

It follows from the last lemma that the expected complexity of the incremental construction algorithm is in $O(n \log n)$.

References

[1] F.Preparata and M.I. Shamos. *Computational Geometry: An Introduction*, Springer Verlag 1985.

[2] A. Okabe, B. Boots and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley, 2nd edition, 2000.

[3] F. Aurenhammer. Voronoi Diagrams: A survey of a fundamental geometric data structure, ACM Computing Surveys, 23(3):345-405, Sept. 1991.