Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_507-1

Voronoi Diagrams and Delaunay Triangulations

Rolf Klein*
Institute for Computer Science, University of Bonn, Bonn, Germany

Keywords Computational geometry • Delaunay triangulation • Distance problem • Edge flip • Minimum spanning tree • Sweepline • Voronoi diagram

Years and Authors of Summarized Original Work

1975; Shamos, Hoey
1987; Fortune

Problem Definition

Suppose there is some set of objects p called *sites* that exert influence over their surrounding space, M. For each site p, we consider the set of all points z in M for which the influence of p is strongest.

Such decompositions have already been considered by R. Descartes [5] for the fixed stars in solar space. In mathematics and computer science, they are called *Voronoi diagrams*, honoring work by G.F. Voronoi on quadratic forms. Other sciences know them as *domains of action, Johnson-Mehl model, Thiessen polygons, Wigner-Seitz zones*, or *medial axis transform*.

In the case most frequently studied, the space M is the real plane, the sites are n points, and influence corresponds to proximity in the Euclidean metric, so that the points most strongly influenced by site p are those for which p is the nearest neighbor among all sites. They form a convex region called the *Voronoi region* of p. The common boundary of two adjacent regions of p and q is a segment of their *bisector* $B(p, q)$, the locus of all points of equal distance to p and q. An example of 10 point sites is depicted in Fig. 1.

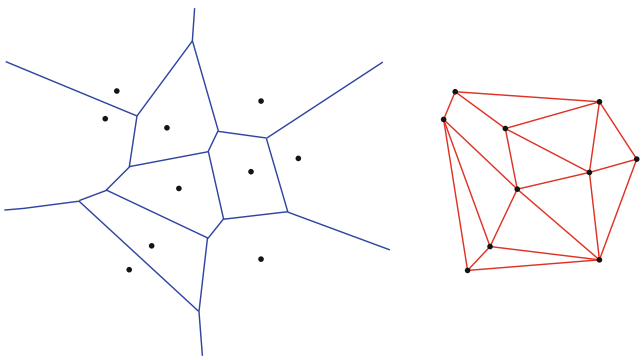

Fig. 1 Voronoi diagram and Delaunay triangulation of 10 point sites in the Euclidean plane

*E-mail: rklein@uni-bonn.de, rolf.klein@uni-bonn.de

DOI 10.1007/978-3-642-27848-8_507-1

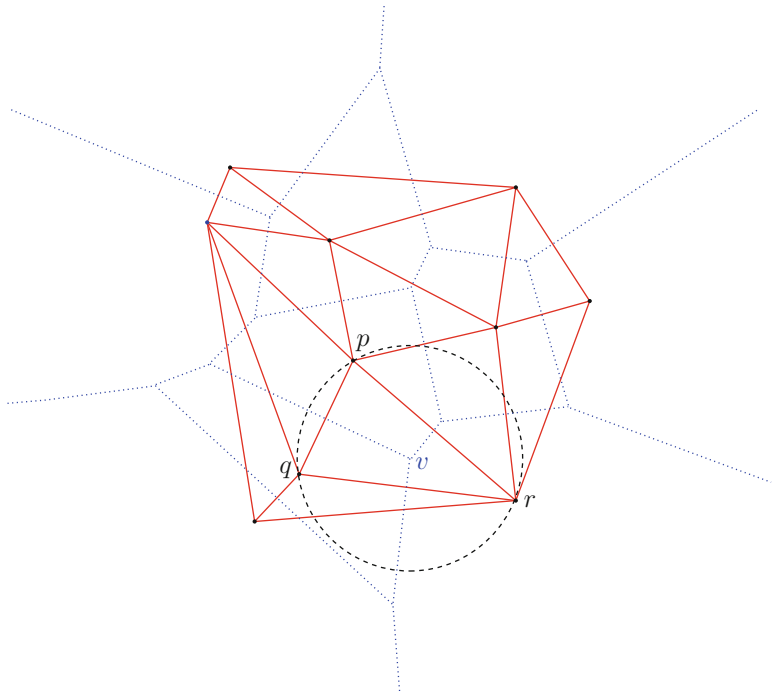

Fig. 2 The empty circle property

Let us assume that the set S of point sites is in general position, so that no three points are situated on a line, and no four on a circle. Then the Voronoi diagram $V(S)$ of S is a connected planar graph. Its vertices are those points in the plane which have three nearest neighbors in S, while the interior edge points have two. As a consequence of the Euler formula, $V(S)$ has only $O(n)$ many edges and vertices.

If we connect with line segments, those sites in S whose Voronoi regions share an edge in $V(S)$, a triangulation $D(S)$ of S results, called the *Delaunay triangulation* or *Dirichlet tessellation*; see Fig. 1. Each triangle with vertices p, q, r in S is dual to a vertex v of $V(S)$ situated on the boundary of the Voronoi regions of p, q, and r. Because p, q, r are the nearest neighbors of v in S, the circle through p, q, r centered at v contains no other point of S. Thus, $D(S)$ consists of triangles with vertices in S whose circumcircles are empty of points in S; see Fig. 2. Conversely, each triangle with empty circumcircle occurs in $D(S)$.

Given a set S of n point sites, the problem is to efficiently construct one of $V(S)$ or $D(S)$; the dual structure can then easily be obtained in linear time.

Generalizations

Voronoi diagrams can be generalized in several ways. Instead of point sites, other geometric objects can be considered. One can replace the Euclidean distance with distance measures more suitable to model a given situation. Instead of forming regions of all points that have the same nearest site, one can consider higher-order Voronoi diagrams where all points share a region for which the nearest k sites are the same, for some k between 2 and $n-1$. Many more variants can be found in [9] and [1]. Abstract Voronoi diagrams provide a unifying framework for some of the variants mentioned; see the corresponding chapter in this encyclopedia.

DOI 10.1007/978-3-642-27848-8_507-1

Key Results

Quite a few algorithms for constructing the Voronoi diagram or the Delaunay triangulation of n points in the Euclidean plane have been developed.

Divide and Conquer

The first algorithm was presented in the seminal paper [11], which gave birth to the field of computational geometry. It applies the *divide and conquer* paradigm. Site set S is split by a line into subsets L and R of equal cardinality. After recursively computing $V(L)$ and $V(R)$, one needs to compute the bisector $B(L, R)$, the locus of all points in the plane that have a nearest neighbor in L and in R. This bisector is an unbounded monotone polygonal chain. In time $O(n)$ one can find a starting segment of $B(L, R)$ at infinity, and trace the chain through $V(L)$ and $V(R)$ simultaneously. Thus, the algorithm runs in time $O(n \log n)$ and linear space, which is optimal.

Sweep

How to design a left-to-right *sweepline* algorithm for constructing $V(S)$ is not obvious. When the advancing sweepline H enters the Voronoi region of p before site p has been detected, it is not clear how to correctly maintain the Voronoi diagram along H. This difficulty has been overcome in [7] by applying a transformation that ensures that each site is the leftmost point of its Voronoi region. In [10] and [4], a more direct version of this approach was suggested. At each time during the sweep, one maintains the Voronoi diagram of all point sites to the left of sweepline H, and of H itself, which is considered a site of its own; see Fig. 3. Because the bisector of a point and a line is a parabola, the Voronoi region of H is bounded by a connected chain of parabolic segments,

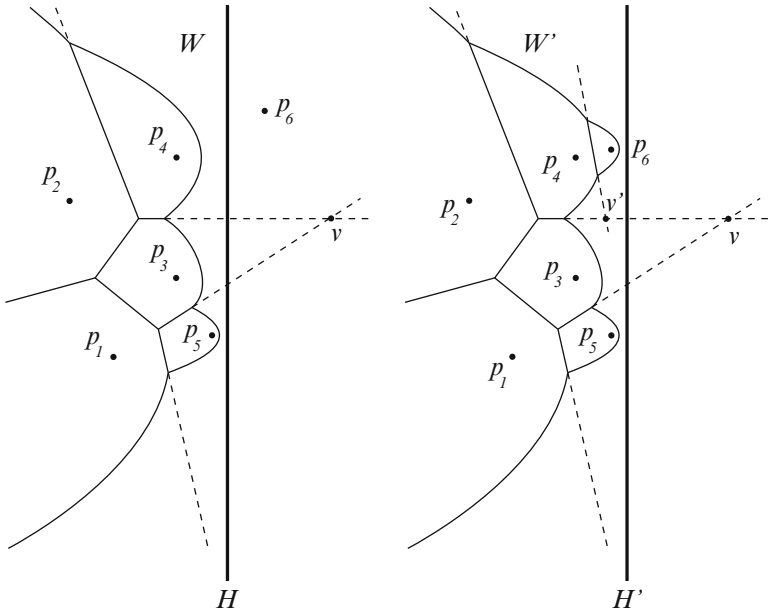

Fig. 3 The sweepline advancing to the right

DOI 10.1007/978-3-642-27848-8_507-1

called the *wavefront* W. As H moves to the right, W follows at half the speed. Each point z to the left of W is closer to some point site p left of H than to H and, all the more, to all point sites to the right of H that are yet to be discovered. Thus, the Voronoi regions of the point sites to the left of W keep growing, as sweepline H proceeds, along the extensions of Voronoi edges beyond W; these *spikes* are depicted by dashed lines in Fig. 3.

There are two kinds of events one needs to handle during the sweep. When sweepline H hits a new point site (like point p_6 in Fig. 3), a new wave separating this point site from H must be added to W. When wavefront W hits the intersection of two neighboring spikes, the wave between them must be removed from W; this will first happen in Fig. 3 when W' arrives at v'. Intersections between neighboring spikes can be determined as in the standard line segment intersection algorithm [2]. There are only $O(n)$ many events, one for each point site and one for each Voronoi vertex v of $V(S)$. Since wavefront W is always of linear size, the sweepline algorithm runs in $O(n \log n)$ time using linear space.

Reduction to Convex Hull

A rather different approach [3] obtains the Delaunay triangulation in dimension 2 from the convex hull in dimension 3, which can itself be constructed in time $O(n \log n)$. As suggested in [6], one vertically lifts the point sites to the paraboloid $Z = X^2 + Y^2$ in 3-space. The lower convex hull of the lifted points, projected onto the XY-plane, equals the Delaunay triangulation, $D(S)$.

Incremental Construction

Another, very intuitive algorithm first suggested in [8] constructs the Delaunay triangulation incrementally. In order to insert a new point site p_i into an existing Delaunay triangulation $D(S_{i-1})$, one first finds the triangle containing p_i and connects p_i to its vertices by line segments. Should p_i be contained in the circumcircles of adjacent triangles, the Delaunay property must be restored by *edge flips* that replace the common edge of two adjacent triangles T, T' by the other diagonal of the convex quadrilateral formed by T and T'. If the insertion sequence of the p_i is randomly chosen, a running time in $O(n \log n)$ can be expected. Details on all algorithms can be found in [1].

Applications

Although of linear size, Voronoi diagram and Delaunay triangulation contain a lot of information on the point set S. Once $V(S)$ or $D(S)$ are available, quite a few distance problems can be solved very efficiently. We mention only the most basic applications here and refer to [1] and [9] for further reading.

By definition, the Voronoi diagram reduces the *post office* or *nearest neighbor* problem to a point location problem: given an arbitrary query point z, the site in S nearest to z can be found by determining the Voronoi region containing z. In order to find the *largest empty circle* whose center z lies inside a convex polygon C over m vertices, one needs to inspect only three types of candidates for z, the vertices of $V(S)$, the intersections of the edges of $V(S)$ with the boundary of C, and the vertices of C. All these can be done in time $O(n + m)$.

DOI 10.1007/978-3-642-27848-8_507-1

If the site set S is split into subsets L and R, then the closest pair $p \in L$ and $q \in R$ forms an edge of the Delaunay triangulation $D(S)$ (which crosses the Voronoi edge separating the regions of p and q). This fact has nice consequences. First, the nearest neighbor of a site $p \in S$ must be one of its neighboring vertices in $D(S)$. Hence, *all nearest neighbors* and the *closest pair* in S can be found in linear time once $D(S)$ is available, because $D(S)$ has only $O(n)$ many edges. Second, $D(S)$ contains the *minimum spanning tree* of S, which can be extracted from $D(S)$ in linear time.

Remarkable and useful is the *equiangularity property* of $D(S)$. Of all (exponentially many) triangulations of S, the Delaunay triangulation maximizes the ascending sequence of angles occurring in the triangles, with respect to lexicographic order. In particular, the minimum angle is as large as possible. In fact, if a triangulation is not Delaunay, it must contain two adjacent triangles, such that the circumcircle of one contains the third vertex of the other. By flipping their common edge, a new triangulation with larger angles is obtained.

Cross-References

- 3D Conforming Delaunay Triangulation
- Abstract Voronoi Diagrams
- Delaunay Triangulation and Randomized Constructions
- Optimal Triangulation

Recommended Reading

1. Aurenhammer F, Klein R, Lee DT (2013) Voronoi diagrams and delaunay triangulations. World Scientific, Singapore
2. Bentley J, Ottman T (1979) Algorithms for reporting and counting geometric intersections. IEEE Trans Comput C-28:643–647
3. Brown KQ (1979) Voronoi diagrams from Convex Hulls. Inf Process Lett 9:223–228
4. Cole R (1989) as reported by ÓDúnlaing, oral communication
5. Descartes R (1644) Principia philosophiae. Ludovicus Elsevirius, Amsterdam
6. Edelsbrunner H, Seidel R (1986) Voronoi diagrams and arrangements. Discret Comput Geom 1:25–44
7. Fortune S (1978) A sweepline algorithm for Voronoi diagrams. Algorithmica 2:153–174
8. Green PJ, Sibson RR (1978) Computing dirichlet tessellations in the plane. Comput J 21: 168–173
9. Okabe A, Boots B, Sugihara K, Chiu SN (2000) Spatial tessellations: concepts and applications of Voronoi diagrams. Wiley, Chichester
10. Seidel R (1988) Constrained delaunay triangulations and Voronoi diagrams with obstacles. Technical report 260, TU Graz
11. Shamos MI, Hoey D (1975) Closest-point problems. In: Proceedings 16th annual IEEE symposium on foundations of computer science, Berkeley, pp 151–162