Ch. 7 Voronoi Diagrams

Let \( P = \{ p_1, p_2, \ldots, p_n \} \) be \( n \) pts in the plane.

The Voronoi diagram of \( P \), \( \text{Vor}(P) \) is the subdivision of the plane into \( n \) cells, one for each \( p_i \in P \).

- cell \( V(p_i) = \{ q \mid \text{dist}(p_i, q) < \text{dist}(p_j, q) \ \forall p_j \in P, p_j \neq p_i \} \)

![Diagram](diagram)

Set of points closer to \( p_i \) than to \( p_2, p_3 \) or \( p_4 \).

**Cell** \( V(p_i) \)

- defined by intersect of halfplanes \( h(p_i, p_j), p_j \neq p_i \)
- where \( h(p_i, p_j) \) is open halfplane containing \( p_i \) and bounded by perpendicular bisector of \( p_i \) and \( p_j \)
- \( h(p_i, p_j) \) contains all pts closer to \( p_i \) than to \( p_j \)

\[
V(p_i) = \bigcap_{k \neq i} h(p_i, p_k)
\]

- \( V(p_i) \) is convex polygonal region bounded by at most \( n-1 \) vertices + \( n-1 \) edges

**Theorem 7.2** Let \( P \) be set of \( n \) points in plane.

If all sites are collinear, then \( \text{Vor}(P) \) is \( n-1 \) parallel lines + \( n \) cells.

Else, \( \text{Vor}(P) \) is connected + its edges are either segments or half-lines.
Complexity of Vor(P) (#verts + #edges)
* each Voronoi cell has at most $n-1$ edges + $n-1$ vertices
  - Size of Vor(P) is at most quadratic

Is this pessimistic? Yes...

Theorem 7.3 The number of vertices of Vor(P) is at most $2n-5$ and the number of edges is at most $3n-6$.

Proof
Trivially true if sites are collinear. Assume they are not.
* We'll use Euler's Formula for connected planar graphs:
  \[ v - e + f = 2 \]
  \#verts \#edges \#faces
* We cannot directly apply Euler's Formula since Vor(P) has infinite (half-line) edges.
  - add a vertex $v_\infty$ "at infinity" & connect all half-infinite edges to $v_\infty$
  - Vor($P \cup v_\infty$) has $(n_v + 1) - e_v + f_v = 2$
  - $n_v + 1 - e_v + 2 = 2$ \[ n_v + 1 - e_v + 2 = 2 \]
  - $n_v + 1 = e_v$ \[ n_v + 1 = e_v \]
  - $2n_v = \text{sum of degrees of all vertices in Vor}(P \cup v_\infty)$ \[ 2n_v = \sum \text{of degrees of all vertices in Vor}(P \cup v_\infty) \]
  - every vertex has degree at least 3
  \[ 2e_v \geq 3(n_v + 1) \]

* $3n_v + 3 \leq 2n_e$ \[ n_v \leq 2n - 5 \]
  - by (3) \[ n_v \leq 2n - 5 \]
* $3n_v + 3 \leq 2(n_v + n - 1)$ \[ n_v \leq n_v + n - 1 \]
  - by (1) \[ n_v \leq n_v + n - 1 \]
* $n_v \leq 2n - 5$ \[ n_v \leq 2n - 5 \]
  - algebra
* $n_e = n_v + n - 1$ \[ n_e = n_v + n - 1 \]
  - by (1)
* $n_e \leq (2n-5) + n - 1$ \[ n_e \leq (2n-5) + n - 1 \]
  - by (3)
* $n_e \leq 3n - 6$ \[ n_e \leq 3n - 6 \]
  - algebra
Characterization of edges $\&$ vertices of $\text{Vor}(P)$
- edges: parts of bisectors between sites
- vertices: intersection pts of bisectors

but, are $\Theta(n^2)$ bisectors $\&$ not all define edges...

**Def:** The largest empty circle of $q$ wrt $P$ $C_p(q)$ is largest circle w/ center $q$ that doesn't contain any $p \in P$ in its interior.

**Theorem 7.4** For $\text{Vor}(P)$ of pt set $P$:
(i) $q$ is a vertex of $\text{Vor}(P) \iff C_p(q)$ has 3 or more sites on its boundary (no one in interior)
(ii) The bisector of sites $p_i + p_j$ defines edge of $\text{Vor}(P)$ \iff there is $q \in \mathbb{R}^2$ s.t. $C_p(q)$ contains $p_i + p_j$ on its boundary $\&$ no other sites on boundary or inside.

**Proof**
(i) $\iff$ Let $q$ be pt w/ $C_q(P)$ having $p_i, p_j, p_k$ on boundary.
$\triangleright$ Since $C_q(P)$ is empty, $q$ must be on boundary of $V(p_i), V(p_j), V(p_k)$ $\iff$ a vertex of $\text{Vor}(P)$
($\Rightarrow$) every vertex $q$ of $\text{Vor}(P)$ is incident to at least 3 Voronoi cells $V(p_i), V(p_j), V(p_k)$
* $q$ is equidistant $p_i, p_j, p_k$ $\&$ no closer site since $V(p_i), V(p_j), V(p_k)$ meet at $q$.
$\triangleright$ hence interior of circle w/ $p_i, p_j, p_k$ on boundary must be empty.

(ii) is similar.
Computing $\text{Vor}(P)$

**Naive Approach**
- compute $V(p_i) = \bigcap_{j \in i} h(p_i, p_j)$ for each $p_i \in P$
- use halfplane intersection algorithm - $O(n \log n)$ time each $V(p_i)$
  $\Rightarrow O(n^2 \log n)$

**Fortune's Sweepline Algorithm** - $O(n \log n)$
- sweep horizontal sweepline from top to bottom
  - instead of maintaining intersection of $\text{Vor}(P) \cup I$
  - keep track of portion of $\text{Vor}(P)$ above $I$ that is computed already (cannot be changed by sites below $I$)
  - the beachline

What part of $\text{Vor}(P)$ above $I$ cannot be changed anymore?
- region of $\mathbb{R}^2$ that is closer to some $p_i \in P$ above $I$ than to $I$
  - for each such $p_i$ above $I$, these pts closer to $p_i$ than $I$ are bounded by a parabola
  - set of all such pts bounded by parabolic arcs
  $\Rightarrow$ beachline
**Fact** The beach line is $x$-monotone (intersected by a vertical line in exactly one point).

**Note:** Breakpoints between parabolic arcs lie on edges of $\text{Vor}(P)$ \Rightarrow breakpoints trace out $\text{Vor}(P)$ as sweepline moves

**Sweepline Status** \Rightarrow beachline

**Event points**
1. When new arc appears in beachline (site event)
2. When arc disappears from beachline (circle event)

**Site event** (new arc appears)

- When new "arc" appears it is just a pt on beach line
  - Two new breakpoints appear that trace out an edge of $\text{Vor}(P)$
    (move in opposite directions)

**Fact** Beach line consists of at most $2n-1$ parabolic arcs
- Each site adds one arc
- Splits an existing arc in two (all except first)
Circle event (an arc disappears)
- an existing arc shrinks to a point & disappears

\[ \alpha + \alpha'' \text{ cannot be part of same parabola} \]
- when \( \alpha, \alpha', \alpha'' \) coincide we have apt of Vor(P) say \( q \)
- \( q \) is equidistant \( l \) & \( \Pi, \Pi', \Pi'' \)
\[ \Rightarrow \text{circle centered at } q \text{ has lowest pt on } l \text{ must be empty} \]

\[ \Rightarrow \text{when arc disappears} \]
\[ \Rightarrow \text{two breakpoints meet (defining vertex of Vor(P))} \]
\[ \Rightarrow \text{this is lowest point of circle thought sites (corr. to 3 adj arcs) meets sweep line. (event pt)} \]
\[ \Rightarrow \text{circle event (lowest point of circle)} \]

**DATA STRUCTURES**

1. **Vor(P)**
   - doubly connected edge list.
   - add bounding box to sites in Vor(P) to have finite edges

2. **BEACH LINE \( T \)**
   - balanced binary search tree
   - leaves arcs of beach line (left to right order)
   - internal nodes are breakpoint

3. **EVENT QUEUE \( Q \)**
   - sites (ports)
   - circle events (ptrs to arcs in \( T \) which disappear)
     (lowest pt of circle)
Algorithm Voronoi Diagram (P)
1. init Q w/ all site events
2. while (Q not empty)
3. get highest event pt in Q
4. if a SITE-EVENT, then HANDLE-SITE-EVENT
   else HANDLE-CIRCLE-EVENT
5. remove event from Q
6. end while
7. fix up Vor(P) (have half-infinite edges, etc).

HANDLE-SITE-EVENT (pi)
1. search in T for arc a vertically above pi;
   - delete all circle events involving a from Q
2. replace a's leaf in T w/ 3-leaf subtree
   - middle leaf stores pi
   - outer leaves store pt pj corr. to a
   - inner nodes breakpts (pi, pi) + (pi, pj)
3. create new records for Vor(P) for half-edge
   separating V(pi) and V(pj) (traced by new breakpoints)
4. check triples for new arcs
   - insert new circle events if circle intersects 2 and
   -  is not in Q already

[Diagram of Voronoi diagram with points pi, pj, and arc a]
**HANDLE-CIRCLE-EVENT** ($p_0$)

1. Search in $T$ for arc $\alpha$ vertically above $p_0$
   - delete all circle events involving $\alpha$ from $Q$
2. delete leaf representing $\alpha$ from $T$.
   - update tuples representing breakpoints at internal nodes
3. add center of circle as vertex record in Vor($P$)
   - create half-edge records for new breakpoint
4. check triples of arcs
   - insert new circle events to $Q$ if necessary (new + intersect $l$)

---

**Running Time:** $O(n \log n)$
- Handle event pts $O(\log n)$ htree
  - search + rebalance + insert + delete
- $O(n)$ event pts (never insert same pt twice)
  - can "change" deleted circle events to the "true" event we are processing at that time
  - each "circle" event processed in vertex Vor($P$) and there are $O(n)$ of these.

$\Rightarrow$ $O(n \log n)$ total.

**Storage:** $O(n)$
- tree + Vor($P$) + Queue
Degeneracies

1) two or more pts on horizontal line
   - two sites w/ same y-coordinate
     - break ties arbitrarily
   - coinciding event points.
     - 4 co-circular sites
     - ignore them (treat as multiple events)
     - in post-processing "merge" duplicate vertices.

2) site pi vertically below breakpoint
   - not clear which are to split...
   - get zero length arc
     ⇒ it will be removed later (when process circles).

Duality

The Voronoi diagram of Vor(P) in d-dimensions can
be transformed in the computation of a convex hull (intersection
of half-spaces in (d+1)-dimensions.
(details in Ch. 11.5)

e.g.
- map pt \( p = (p_x, p_y) \) to (non-vertical) plane \( z = 2p_x \cdot x + 2p_y \cdot y - (p_x^2 + p_y^2) \)
- intersection of half-spaces gives convex polyhedron
- projection of its verts + edges gives Vor(P).

\[
\begin{align*}
\text{halfspace tangent to paraboloid base.}
\end{align*}
\]
Voronoi Diagram in $d$-dim = Convex Hull in $(d+1)$-dim

Unit Paraboloid in 3-space

\[ U := z = x^2 + y^2 \]

Point on x-y plane ($z = 0$) \( p = (p_x, p_y, 0) \)

Vertical line through \( p \) intersects \( U \) in

\[ p' = (p_x, p_y, p_x^2 + p_y^2) \]

\( h(p) \) is non-vertical plane \( z = 2p_x x + 2p_y y - (p_x^2 + p_y^2) \)

\( h(p) \) contains \( p' \)
The Delaunay Triangulation

Def: The Delaunay triangulation $\Delta$ is the dual graph $G$ of the Voronoi diagram $\text{Vor}(P)$.

$\text{DG}(P)$
- has a node for every Voronoi cell $V(p_i)$
- arc between nodes if corr. cells are adjacent in $V(p_i)$

Note: $\text{DG}(P)$ has an arc for every edge of $\text{Vor}(P)$

Fact: The Delaunay graph of a point set is plane graph.

Fact: If no 4 points of $P$ are co-circular, $\text{DG}(P)$ is a triangulation. (Call point set in general position if no 4 co-circular)

and gives us Delaunay triangulation $\Delta$. If not in general position, we call any triangulation of $\text{DG}(P)$ the Delaunay triangulation.

$\text{DG}(P)$
\text{NOT: a triangulation...}$\text{Vor}(P)$
**Legal Triangulations** (Sect 9.1)

- Let $T$ be a triangulation with $m$ triangles.
- Let $\{x_1, x_2, \ldots, x_m\}$ be the angles of the triangles sorted in increasing order.
  
  $A(T) = (x_1, x_2, \ldots, x_m)$ is the **angle vector** of $T$.

**Def** A triangulation $T$ is **angle optimal** if

$A(T) \geq A(T')$ (lexicographic ordering) for all triangulations $T'$ of $P$.

Angle optimal triangulations are useful for many applications:
- maximize the **minimal angle**

**Theorem 9.9** Let $P$ be a set of points in the plane.

- Any angle optimal triangulation of $P$ is a Delaunay Triangulation.
- Any Delaunay triangulation of $P$ maximizes the minimum angle over all triangulations of $P$.

**Computing Delaunay Triangulations**

1. Can first compute $\text{vor}(P)$ and then construct its dual graph (and triangulate it).

2. Can compute it directly using a randomized incremental algorithm (Sect 9.3)
   - expected time $O(n \log n)$
   - expected storage $O(n)$