Two powerful geometric constructions

Jean-Michel Moreau[1]

[1]LIRIS
Université Claude Bernard, LYON 1.
Jean-Michel.Moreau@liris.univ-lyon1.fr

December 2004 / Fudan U., Shanghai (invitation Pr. Sun Weichi)

Outline

What is computational geometry?

- CG implements real world geometry with rich algorithmic structures and machine arithmetic.
- Simulations, construction, geometric computations / measurement, help in decision.
- Applications: computer graphics, terrain simulation, geography / environment, telephone technology...

Lost in the Alps (c. 2000)

- Three friends on a winter mountain hike in the Alps.
- Storm gathered. Got caught in blizzard.
- They had a cellular telephone with a weak battery. Made a desperate call to friend, battery expired, friend called rescue team, who called the police.
- The telephone company finally managed to locate them. All got rescued (safe), after three long (and *costly*) days of suspense.

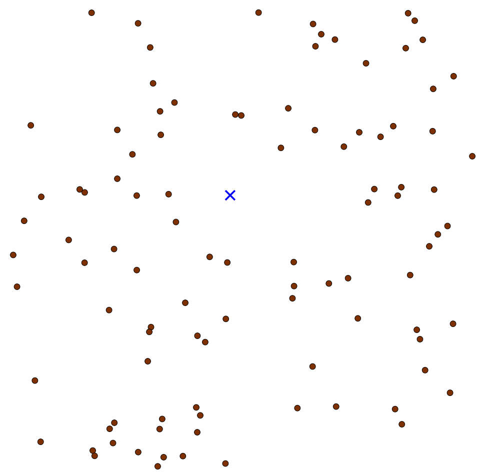
Mathematical formulation

- Data:
  - Let $S_n = \{s_1, s_2, ..., s_n\}$ be a “cloud” of n distinct points (called sites[1]) in the Euclidean plane E^2.
  - Let q be *any* point in E^2.
- Question: What is the site “closest” to q in S_n?
  i.e., find (possibly *not unique*) $j \in [1, n]$ such that:

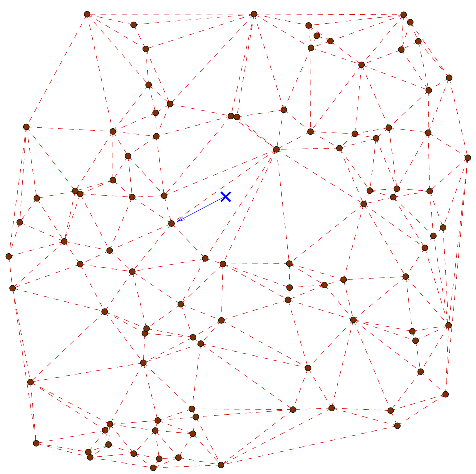
$$d(q, s_j) \leq d(q, s_i) \; \forall i \in [1, n].$$

[1] Here: telephone relays

Example (query and nearest site)

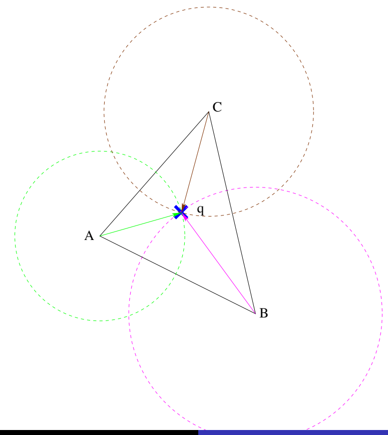


Example (Delaunay mesh)



Final location – geometrical approach

Distances from cell phone to three nearest relays allow computing actual location of phone call.



Final location – analytical approach

$$q \in \Gamma(A, r_A) \Leftrightarrow (x_q - x_A)^2 + (y_q - y_A)^2 = r_A^2 \quad (1)$$
$$q \in \Gamma(B, r_B) \Leftrightarrow (x_q - x_B)^2 + (y_q - y_B)^2 = r_B^2 \quad (2)$$
$$q \in \Gamma(C, r_C) \Leftrightarrow (x_q - x_C)^2 + (y_q - y_C)^2 = r_C^2 \quad (3)$$

Subtract (2) from (1) and (3) from (2) and divide by 2 :

$$x_q(x_B - x_A) + y_q(y_B - y_A) = (r_A^2 - x_A^2 - y_A^2 - (r_B^2 - x_B^2 - y_B^2))/2$$
$$x_q(x_C - x_B) + y_q(y_C - y_B) = (r_B^2 - x_B^2 - y_B^2 - (r_C^2 - x_C^2 - y_C^2))/2$$

Determinant of system on two unknowns (x_q, y_q):

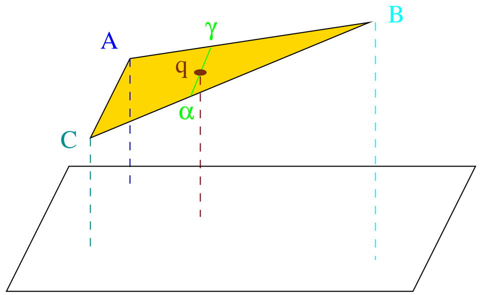
$$\Delta = \begin{vmatrix} x_B - x_A & y_B - y_A \\ x_C - x_B & y_C - y_B \end{vmatrix} \equiv \begin{vmatrix} 1 & 1 & 1 \\ x_A & x_B & x_C \\ y_A & y_B & y_C \end{vmatrix},$$

null if and only if A, B, and C are collinear.

General setting

- A frequent requirement in geo-sciences, but also in other domains (Gouraud shading, etc.)
- Find the value of a given function at a given query point q by interpolating those known at three "sites" surrounding it, say A, B, C.
- Strong constraint: the triangle ABC must not contain any other "site" in its strict interior.

Geographical instantiation: altitudes



$$z_q = (z_\gamma - z_\alpha) \times \overline{\alpha q}/\overline{\alpha\gamma}$$
$$= ((z_B - z_A) \cdot \overline{A\gamma}/\overline{AB} - (z_B - z_C) \cdot \overline{C\alpha}/\overline{CB}) \cdot \overline{\alpha q}/\overline{\alpha\gamma}$$

Limiting special cases

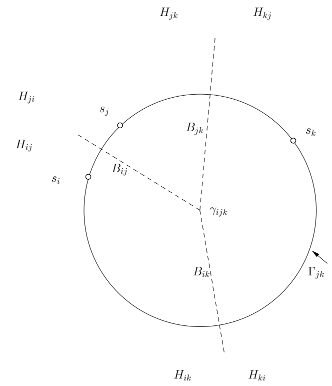
In all these cases, we want to find the site nearest to one query point q, but special cases may arise:

- q coincides with one site.
- q is closest to two (or more!) sites: $\exists\, u, v \in [1, n]^2$ such that $d(q, s_u) = d(q, s_v) \leq d(q, s_i)\ \forall i \in [1, n]$.
- All sites are aligned, and q lies outside supporting line...
- General position assumption:
  1. no more than two sites are aligned;
  2. no more than three sites are co-circular (except if their circumcircle is not empty).

Bisector

- We shall only concentrate on 2-D. Almost all definitions generalize to any dimensions.
- Let $S_n = \{s_1, s_2, ..., s_n\}$ be a set of n sites in general position in $\mathbb{E}^2$.
  Site s_i will be abbreviated to i whenever no confusion is possible.
- Let $(i, j) \in [1, n]^2$, $i \neq j$. $B_{ij} = \{p \in \mathbb{E}^2 \mid d(p, s_i) = d(p, s_j)\}$ is called the bisector of sites i and j.
  Note: $B_{ij} \equiv B_{ji} \, \forall i, j$.
- Let $H_{ij}(H_{ji})$ be the half-plane delimited by and including B_{ij}, and containing $s_i(s_j)$.
  Note: $H_{ij} \cup H_{ji} = \mathbb{E}^2$, $H_{ij} \cap H_{ji} = B_{ij}$.

Bisector – illustration

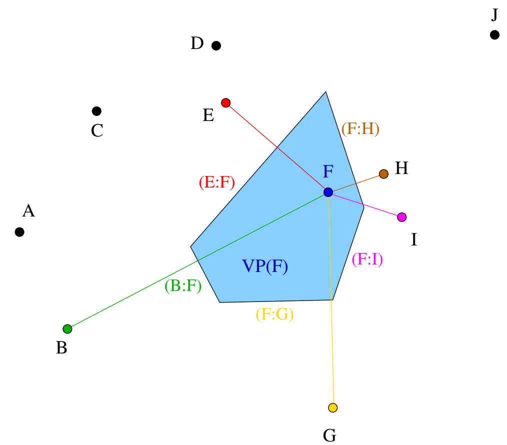


Voronoi polygon – definition

Let $i \in [1, n]$, and s_i be the corresponding site of S_n:

- $\mathcal{PV}(i) = \cap_{j \in [1,n]-\{i\}} H_{ij}$.
- $\mathcal{PV}(i) = \{z \in \mathbb{E}^2 \mid d(z, s_i) \leq d(z, s_j) \, \forall j \in [1, n]\}$.
- All the sites of $S_n - \{s_i\}$ that contribute a bisector with s_i are generically called the nearest neighbours of s_i.

Voronoi polygon – illustration



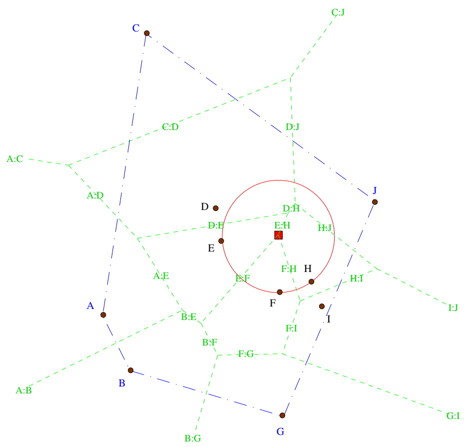
Voronoi polygon – properties

- $s_i \in \mathcal{PV}(i) \implies \mathcal{PV}(i) \neq \emptyset$.
- $\mathcal{PV}(i)$ is convex.
- $\mathcal{PV}(i)$ is unbounded if and only if s_i lies on the boundary of the convex hull of S_n.
- The intersection of the Voronoi polygons of any two distinct sites s_i and s_j is either empty or contains a (possible infinite) section of B_{ij}.
- The intersection of the Voronoi polygons of three distinct sites of S_n is either empty or reduced to a single Voronoi vertex (the center of the circumcircle through these sites).
- Due to the general position assumption, the degree of all Voronoi vertices is exactly three.

Voronoi diagram – definition

- The Voronoi diagram of S_n is the union of all the Voronoi polygons on the sites s_i, $i \in [1, n]$.
- $\mathcal{V}(S_n) = \cup_{i \in [1,n]} \mathcal{PV}(i)$.
- It is a graph whose edges are segments of bisectors on S_n^2, and whose vertices (Voronoi vertices) are the centers of circles circumscribed to triples of sites of S_n.

Voronoi diagram – illustration



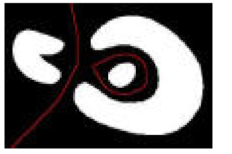
Voronoi diagram – omnipresence in nature (1/2)




Voronoi diagram – omnipresence in nature (2/2)

Voronoi diagram – even in image analysis...

The notion of skiz (skeleton by influence zone)... also known as water shed, dividing line, or great divide.

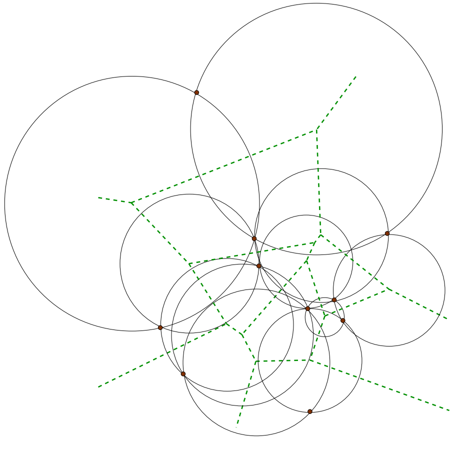


Empty disk property

Theorem

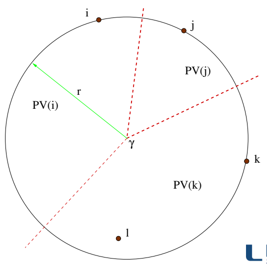
Let s_i, s_j, s_k be three sites of S_n such that $\mathcal{PV}(i) \cap \mathcal{PV}(j) \cap \mathcal{PV}(k) = \{\gamma\}$. The disk centered in γ through s_i, s_j and s_k has no other site in its interior.

Empty disk property – illustration



Empty disk property – proof

- Let s_i, s_j, s_k s.t. $\mathcal{PV}(i) \cap \mathcal{PV}(j) \cap \mathcal{PV}k = \{\gamma\}$.
- $\Gamma(i,j,k)$ circle through s_i, s_j, s_k with center γ, radius $r = d(\gamma, s_i) = d(\gamma, s_j) = d(\gamma, s_k)$.
- Suppose $\Gamma(i,j,k)$ contains a certain site s_ℓ in its interior. Now ℓ is necessarily closer to one of the three sites, say $s_k \implies \ell \in \mathcal{PV}(k)$.
- Then $d(\gamma, l) < r \implies \gamma \in \mathcal{PV}(\ell)$, and not $\mathcal{PV}(i) \cap \mathcal{PV}(j) \cap \mathcal{PV}(k)$, a contradiction.
- Hence $int(\Gamma(i,j,k))$ is necessarily empty.



Introduction
Voronoi diagram
Delaunay triangulation
Definitions
A few examples
Fundamental properties
Construction

Complexity

Theorem

Let S_n contain n distinct sites of $\mathbb{E}^2$. It takes $\Omega(n \log n)$ time to construct the Voronoi diagram of S_n ($\mathcal{VD}(S_n)$).

- Proof:
  - Recall fact: Sorting n real numbers by comparison takes $\Theta(n \log n)$ (optimal) time.
  - Suppose we choose n sites in random order on a line.
  - Build $\mathcal{VD}(S_n)$ in alleged $o(n \log n)$ time.
  - $\mathcal{VD}(S_n) \rightsquigarrow n-1$ parallel lines in increasing order.
  - Hence $\mathcal{VD}(S_n) \rightsquigarrow S_n$ sorted in less than optimal time. A contradiction.

Introduction
Voronoi diagram
Delaunay triangulation
Definitions
A few examples
Fundamental properties
Construction

Divide-and-conquer algorithm (principle) – optimal

To build the Voronoi diagram of S, with size $n \geq 3$ (else, build it 'by hand"):

- Divide step: Split S into two "equal", linearly separable subsets B, R (blue and red / left and right)

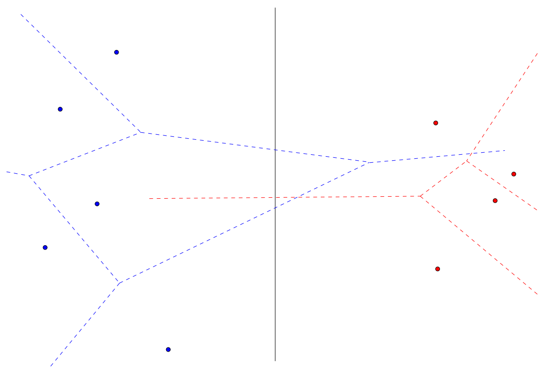
$$(B \cap R = \emptyset, B \cup R = S_n, \ |B| = \lfloor n/2 \rfloor, |R| = n - \lfloor n/2 \rfloor$$

$$\forall\, b \in B, r \in R, b <_{xy} r).$$

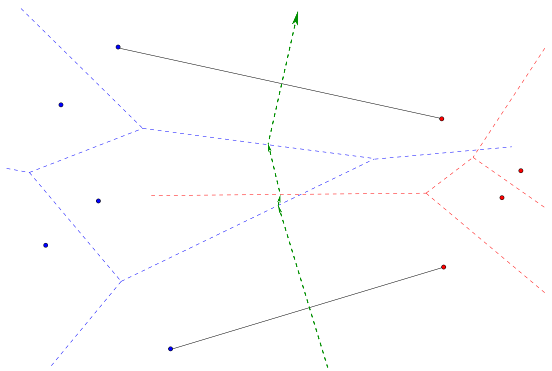
- Recurse on B and R to construct Voronoi diagrams $\mathcal{VD}(B), \mathcal{VD}(R)$.
- Merge step: Follow dividing line at equal distance from one vertex in B and one vertex in R, clipping all edges of $\mathcal{VD}(B)$ that overlap on $\mathcal{VD}(R)$, and conversely.
- Complexity: $O(n \log n)$ (optimal).

Introduction
Voronoi diagram
Delaunay triangulation
Definitions
A few examples
Fundamental properties
Construction

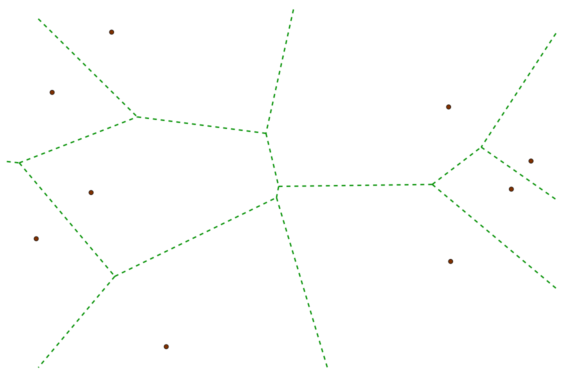
Building $\mathcal{VD}(S)$ divide step + recursion



Building $\mathcal{VD}(S)$ merge step



Building $\mathcal{VD}(S)$ after merge



Voronoi diagram – a summmary

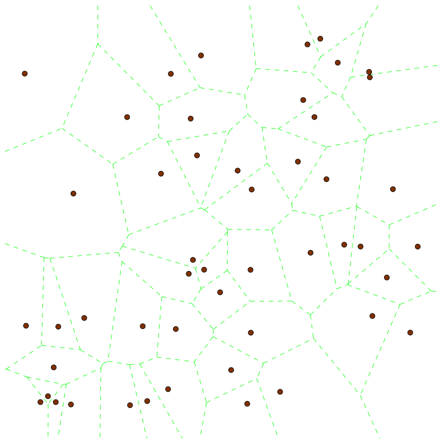
- Although a powerful structure, the Voronoi diagram does not yield a direct answer to the original question : "Which site is nearest to a given query point?"
- To locate the nearest site, we still need to scan all the Voronoi polygons.
- Each search takes optimal $O(\log n)$ time (binary search on convex polygons of size $O(n)$), but there are $O(n)$ such polygons!
- Construction time: $O(n \log n)$. Query time: $O(n) + O(\log n) = O(n)$.
- We need another structure to help us...

Duality

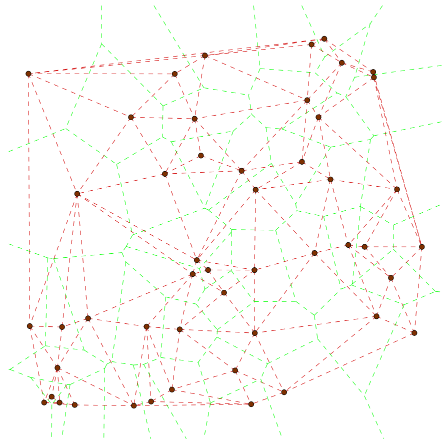
Define the following procedure:

- Connect all couples of sites in S_n whose bisector contributes to $\mathcal{VD}(S_n)$ by a non-null segment.
- The resulting graph is the dual graph of $\mathcal{VD}(S_n)$.
- In 1932, B. Delaunay showed that this graph is a triangulation.
- This graph is called the Delaunay triangulation of S_n.
- By definition, it may be deduced from $\mathcal{VD}(S_n)$ in linear ($O(n)$) time.

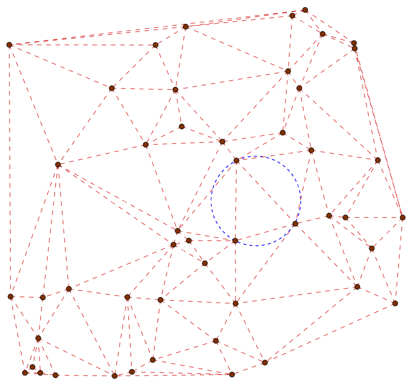
Delaunay triangulation – an illustration



Delaunay triangulation – an illustration



Delaunay triangulation – an illustration



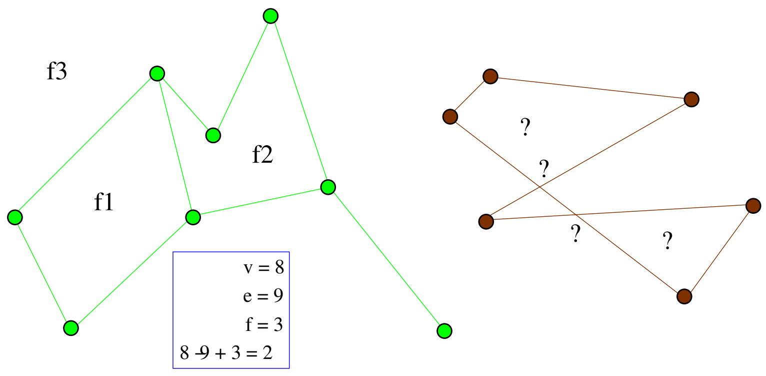
As deduced from $\mathcal{VD}(S_n)$

- The vertices of $\mathcal{DT}(S_n)$ are the sites.
- Its edges are called Delaunay edges, and connect two sites of S_n.
- The edges of $conv(S_n)$ are all Delaunay edges.
- The orthocenter of each Delaunay triangle is a Voronoi vertex of $\mathcal{VD}(S_n)$.
- The circumdisk of triangle $\triangle(s_i, s_j, s_k)$ in $\mathcal{DT}(S_n)$ has no site in its interior.

Counting elements a triangulation – definition

- A graph $\mathcal{G}(V, E)$ is said to be connected if any pair of vertices of V may be linked with (at least) one path (series of edges in E).
- A segment-based graph $\mathcal{G}(V, E)$ is said to be planar if any two edges in E either are disjoint or share *exactly one* endpoint.
- In a planar graph, a face is a cycle (closed path) with no other edge in its interior.
- The external face of a connected planar set is its unbounded face.

Connected planar graph – an illustration



Couting elements in a triangulation – Euler's relation

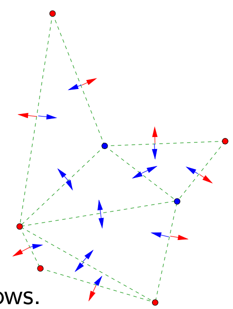
Theorem (L. Euler)

Let $\mathcal{G}(V, E)$ be a planar, connected graph, with f, e, v denoting its number of faces, edges and vertices. Then

$$v - e + f = 2.$$

Counting elements in a triangulation – shooting arrows

Let T be any triangulation in the plane (including $\mathcal{DT}(S_n)$), with t triangles, n vertices and e edges.
Let x be the number of edges on the boundary of its external face X:



- Draw an arrow on both sides of each edge.
- There are two arrows per edge, *i.e.*, $2e$ arrows.
- There are three arrows per triangle, *i.e.*, $3t$ internal arrows.
- There is one outgoing arrow per external face edge, *i.e.*, x external arrows.

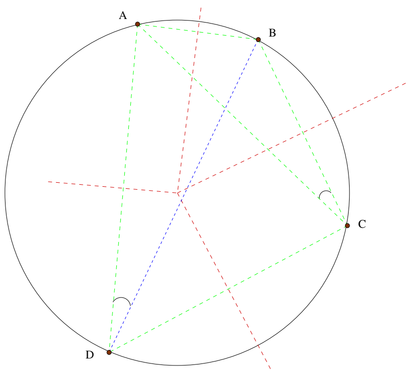
Counting elements in a triangulation – linearity

- Summary:
$$f = t + 1,$$
$$2e = 3t,$$
$$n - e + t = 1 \quad \text{(Euler's relation)}.$$
- This yields:
$$t = 2(n-1) - x,$$
$$e = 3(n-1) - x.$$
- Since $3 \leq x \leq n$:
$$n - 2 \leq t \leq 2n - 5,$$
$$2n - 3 \leq e \leq 3n - 6.$$
- The size of $\mathcal{DT}(S_n)$ (and hence $\mathcal{VD}(S_n)$) is linear.

Empty disk – max min angle equivalence

- Let A, B, C, D be 4 co-circular points, in that order.
- Polygon $Q = (A, B, C, D)$ is convex. Let AB be its smallest edge (w.l.g).
- Then $\langle ADB \rangle = \langle ACB \rangle$ are the smallest of the 12 angles of the two possible triangulations for Q.

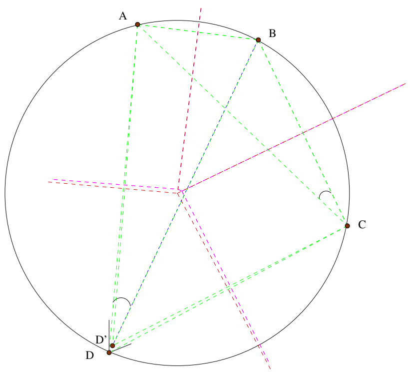
Four circular points – an illustration



Empty disk – max min angle equivalence

- Let D move infinitesimally inside circle.
  1. $\langle ACB \rangle$ remains constant, while $\langle AD'B \rangle > \langle ADB \rangle$.
  2. Disk through A, B, C contains D in its interior.

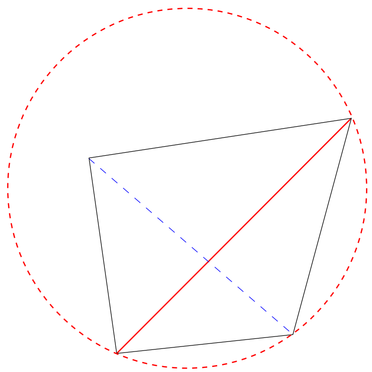
Move D inside – an illustration



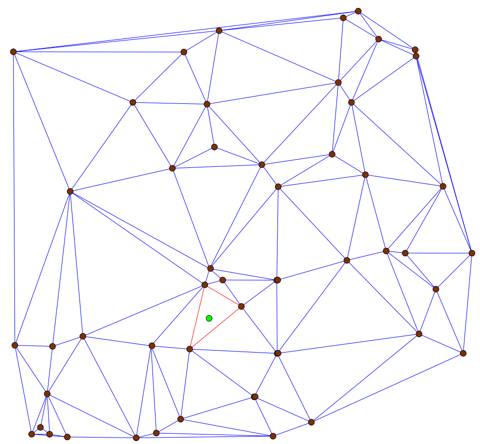
Empty disk – max min angle equivalence

- The empty disk criterion selects diagonal BD.
- Maximizing the minimum angle selects diagonal BD.
- $\rightsquigarrow$ The two criteria (empty disk and maximize minimum angle) are equivalent.
- This gives one constructing method, with two selection equivalent criteria:
  - Start with any triangulation on S_n.
  - While it contains at least one triangle with non-empty circumscribed disk, "flip" diagonals.

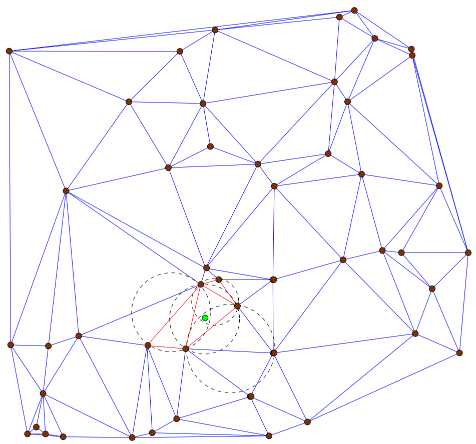
Diagonal "flip" – an illustration



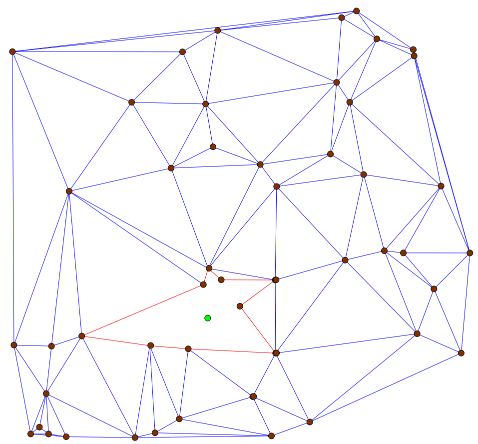
Incremental insertion quadratic running time



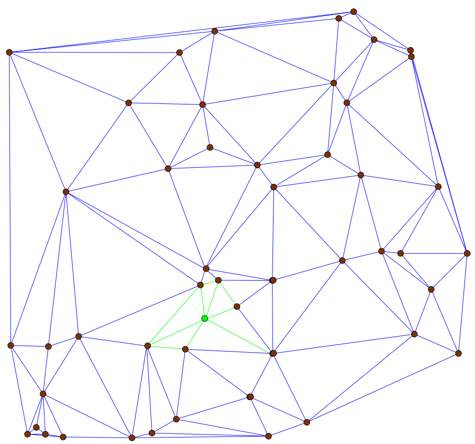
Incremental insertion quadratic running time



Incremental insertion quadratic running time



Incremental insertion quadratic running time



Divide-and-conquer algorithm (principle) – optimal

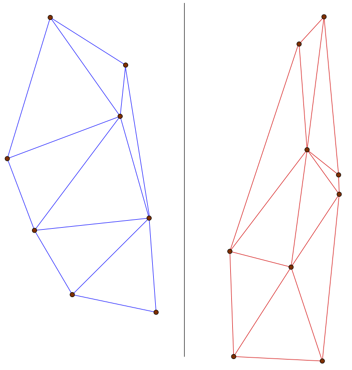
To build the Delaunay triangulation of S, with size $n \leq 3$ (else: build it "by hand"):

- Divide step: Split S into two "equal", linearly separable subsets B, R (blue and red / left and right)

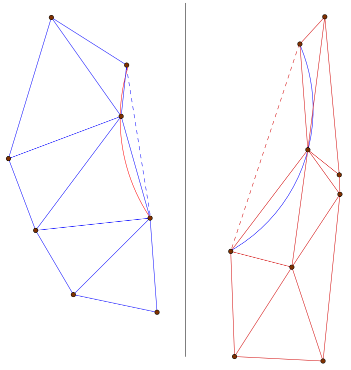
$$(B \cap R = \emptyset, B \cup R = S_n, \ |B| = \lfloor n/2 \rfloor, |R| = n - \lfloor n/2 \rfloor$$
$$\forall\, b \in B, r \in R, b <_{xy} r).$$

- Recurse on B and R to construct Delaunay triangulations $\mathcal{DT}(B), \mathcal{DT}(R)$.
- Merge step: Remove all triangles in $\mathcal{DT}(B)$ whose circumdisk contains a site in R; remove all triangles in $\mathcal{DT}(R)$ whose circumdisk contains a site in B. Fill the "gap" between the two resultant graphs by edges with one vertex in B and the other in R.
- Complexity: $O(n \log n)$ (optimal).

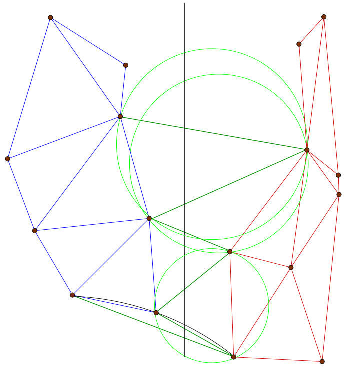
Building $\mathcal{DT}(S)$ divide step + recursion



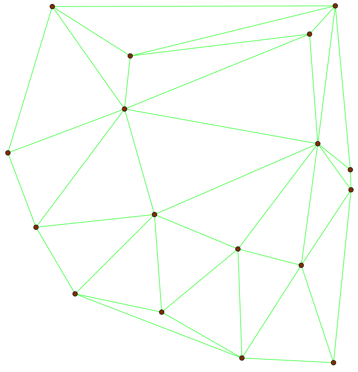
Building $\mathcal{DT}(S)$ merge step : remove conflicting triangles



Building $\mathcal{DT}(S)$ merge step : rebuild



Building $\mathcal{DT}(S)$ after merge



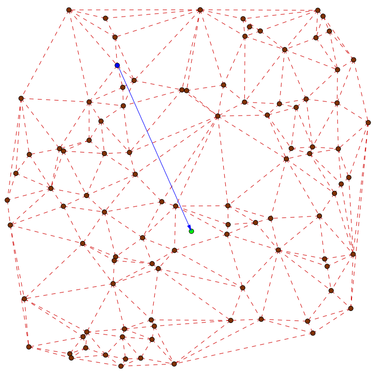
A brute-force solution

- For each triangle t in $\mathcal{DT}(S_n)$:
  - If t contains q: stop
  - Else: $t \leftarrow \texttt{succ}(t)$.
- Complexity: $O(2n-5) = O(n)$.

A finer solution – Jump and walk

- If q lies outside $conv(S_n)$: stop ($O(\log n)$ time).
- Else: pick up a random site s in S_n.
- Find triangle t around s containing segment $[sq]$.
- Walk: until end of time
  - If t contains q: stop
  - Else: $t \leftarrow \texttt{adj}(t)$ % walk to adjacent triangle crossing (sq).
- Complexity: if the sites are uniformly distributed, $O(\sqrt{n})$ on average. [Sibson & Green – proof : Devroye].

Jump and walk – an illustration



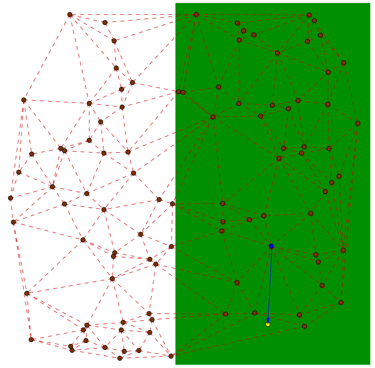
Refined Jump and walk

- If q lies outside $conv(S_n)$: stop ($O(\log n)$ time).
- Else: pick up $k \in [1, n]$ random sites S_n.
- Let s be the site among those closest to q.
- Find triangle t around s containing segment $[sq]$.
- Walk: until end of time:
  - If t contains q: stop
  - Else: $t \leftarrow \texttt{adj}(t)$ % walk to adjacent triangle crossing (sq).
- Complexity: if the sites are uniformly distributed, $O(k + \sqrt{n/k})$ on average. Optimal $O(\sqrt[3]{n})$ reached for $k = \Theta(\sqrt[3]{n})$[Devroye, Zhu].

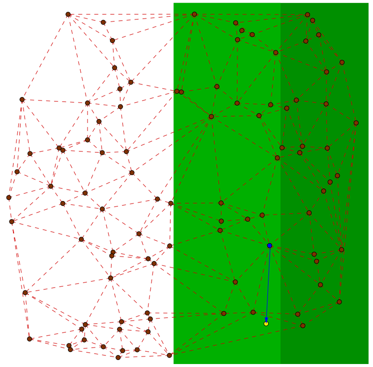
Binsearch and walk

- If q lies outside $conv(S_n)$: stop ($O(\log n)$ time).
- Else: maintain a balanced tree on xy order.
- Search for q in tree and stop when there are $O(k)$ elements in remaining subtree.
- Traverse entire subtree and determine site s closest to q.
- Find triangle t around s containing segment $[sq]$.
- Walk: until end of time
  - If t contains q: stop
  - Else: $t \leftarrow \texttt{adj}(t)$ % walk to adjacent triangle crossing (sq).
- Complexity: if the sites are uniformly distributed, $O(k + \sqrt{n}/k)$ on average. Optimal $O(\sqrt[4]{n})$ reached for $k = \sqrt[4]{n}$. [Devroye, Lemaire, Moreau, 2004].

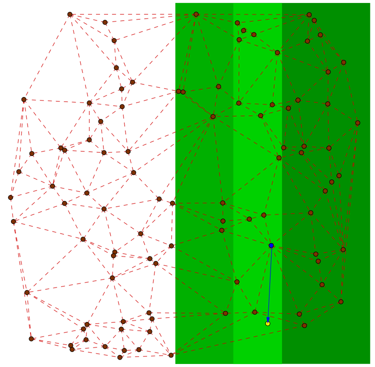
Binsearch and walk – an illustration (1/4)



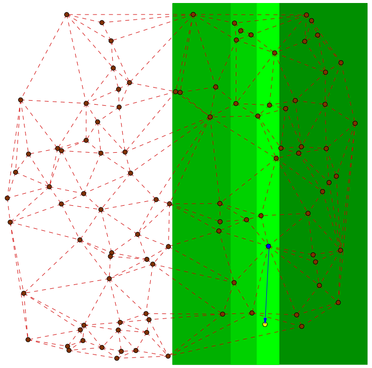
Binsearch and walk – an illustration (1/4)



Binsearch and walk – an illustration (3/4)



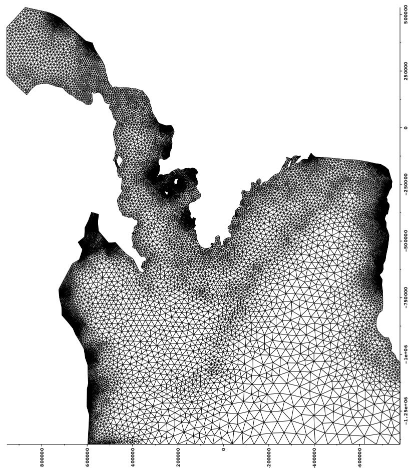
Binsearch and walk – an illustration (4/4)



Constrained adaptive DT (1/2)

- The data set may be a graph, with constrained edges: edges that may not be crossed by any element of the resulting Delaunay triangulation.
  - $\rightsquigarrow$ modify the definition so it allows circumdisks to contain sites that not visible from all three triangle vertices at the same time. Divide-and-conquer becomes much more difficult (JMM 1993).
- The density of the sites may vary with specific parameters. For instance with $\sqrt{h}$, for bathymetric data.
  - $\rightsquigarrow$ apply locally perturbed distribution functionals.

Constrained adaptive DT (2/2) – illustration



Terrain simulation

- Given: a large number of terrain data (x, y, z in 2-D).
- Wanted: a mesh with minimum number of facets and maximum "goodness of fit".
- Method:
  1. Start with initial square (or triangle) covering all data; triangulate with initial diagonal; assign each data point to appropriate triangle.
  2. Repeat until gof reached or all points processed:
     1. Find triangle t containing point p with maximum vertical deflection.
     2. Destroy t and replace with 3 new triangles sharing p.
     3. Redistribute points in t to 3 new triangles.
     4. Reinstaure Delaunay property and redistribute as needed.

Robustness issues

- Geometry with doubles is very different from real geometry.
- Example: Let Ω be the intersection of integral lines AB and CD. Check that $\Omega \in AB$ or CD is next to impossible, due to real number representation in machine.
- One good solution: lazy arithmetic. See explanations on blackboard...