Voronoi Diagrams

Swami Sarvottamananda

Ramakrishna Mission Vivekananda University

NIT-IGGA, 2010

Outline I

1 Introduction
- Motivation for Voronoi Diagram
- Historical Notes and Web References

2 Voronoi Diagrams Concepts
- Definitions
- Formal Definition of Voronoi Diagram

3 Computing Voronoi Diagrams
- Non-optimal Algorithms
- Voronoi Diagram from 3D Convex Hulls
- Fortune's Algorithm
- Divide and Conquer

4 Delaunay Triangulations
- Motivation
- Concepts
- Method of Computation

Outline II

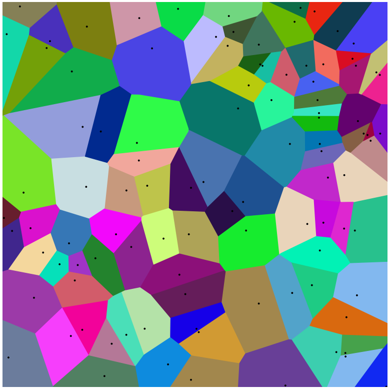
5 Generalisations
- Voronoi Diagrams in $\Re^d$
- Voronoi Diagram for Farthest Points
- Voronoi Diagram for Subsets

6 Conclusion
- Open Problems
- Concluding Remarks
- Further Reading

Voronoi Diagrams

Voronoid Diagram

Example (What does it look like?)



Each cell is a set of points whose nearest site is the same

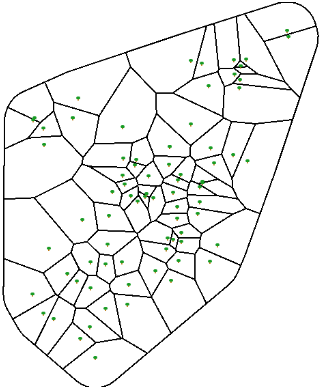
Applications in Cross-disciplines

This concept has independently emerged, and proven useful, in various fields of sciences

- *Medial axis transform* in biology and physiology,
- *Wigner-Seitz zones* in chemistry and physics,
- *Domains of action* in crystallography,
- and *Thiessen polygons* in meteorology and geography.

Other Direct Applications—I

Example (Jurisdiction of Schools in Boston)



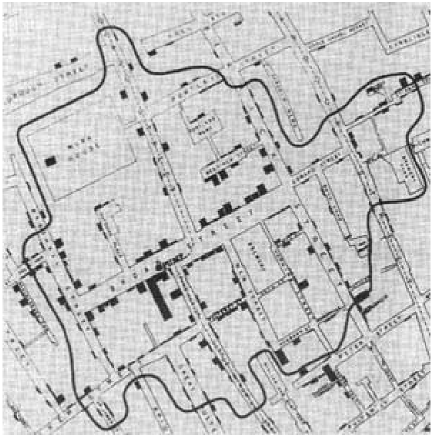
A relevant modern day use.

Other Direct Applications—II

Example (1854 Cholera Epidemic in London)

A particularly notable use of a Voronoi diagram was the analysis of the 1854 cholera epidemic in London, in which physician John Snow determined a strong correlation of deaths with proximity to a particular (and infected) water pump on Broad Street.

John Snow's Voronoi Diagram



Anyone can now guess the infected pump.

Motivation

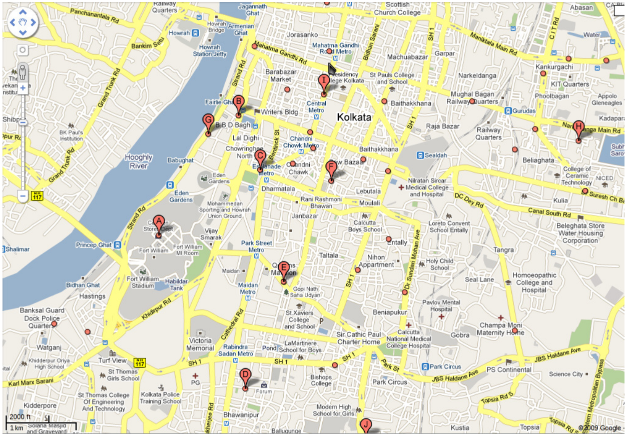
Problem (Voronoi Diagrams — The Post Office Problem)

In a city with several post offices we would like to mark the service region of each post office by proximity. What are these regions?

This problem is mentioned in Chapter 7 of the computational geometry book by de Berg et al. [3].

Let us solve the problem for a section of Kolkata.

Post-offices in a section of Kolkata



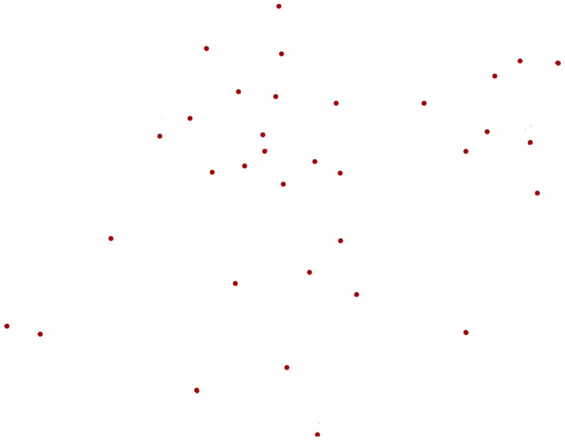
Courtesy: Google Images

Postoffice Problem and its Simplification

We would like to demarcate the service regions of the post-offices.

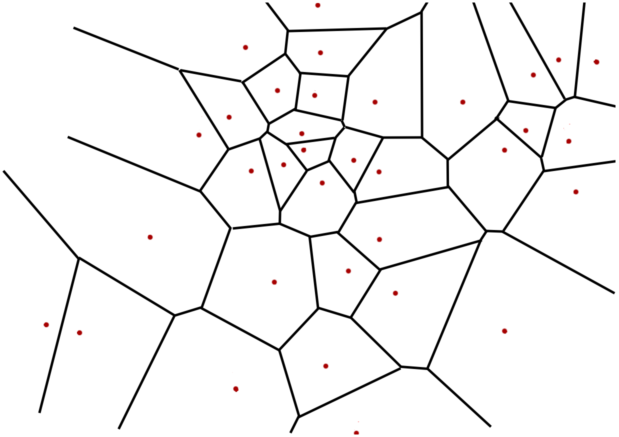
To simplify, let us use the Euclidean distance (how the crows fly) and not the actual distance by the roads.
Secondly, post offices are considered as points.

Post-offices as Points in Plane



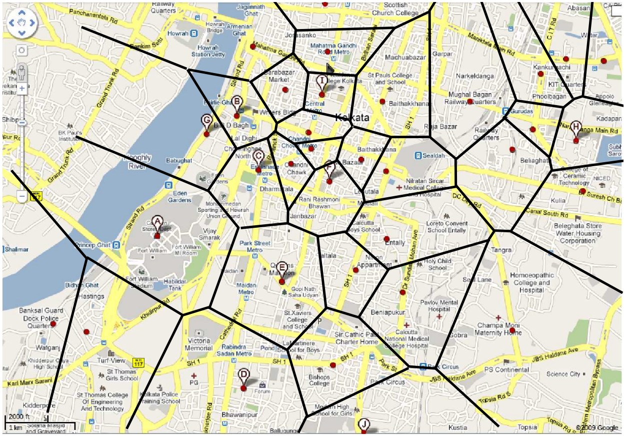
First, we remove the clutter.

Proximity Regions of Postoffices



Second, we compute the Voronoi Diagram

Postoffice Services in Kolkata



Last, we superimpose the solution. Gives correct answer for at least Swami Vivekananda's Birth Place at Simla.

Voronoi Diagrams in Internet

Where you can play around with Voronoi Diagrams—

- Java application at *http://www.nirarebakun.com/voro/ehivorocli.html* (includes higher order VD's),
- VoroGlide (another Java application) *http://wwwpi6.fernuni-hagen.de/GeomLab/VoroGlide/*
- Anything related to Computational Geometry, start at Jeff Erickson's web page, *http://compgeom.cs.uiuc.edu/ jeffe/compgeom/compgeom.html*
- And, much more.

(All the hyperlinks were alive in January 2010, I checked personally)

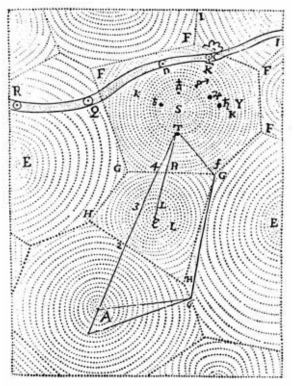
Origin of Problem

The problem is centuries old

- Earliest record in *Principles of Philosophy* [4] by R. Descartes, 17th century.
- Decomposition of space into convex regions of influence of heavenly bodies.

Origin of Problem

Example (Descartes Explanation of Gravity)



Of course gravity does not work like this, but Newton and Einstien were to come later.

What are Voronoi Diagrams

- What are *Voronoi Diagrams*?

What are Voronoi Diagrams

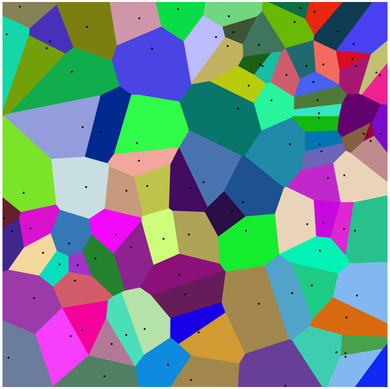
- What are *Voronoi Diagrams*?
- Informally, Voronoi Diagrams are proximity regions to a set of objects. We collect all the points in a set for whom, some particulat proximity query gives the same result.

What are Voronoi Diagrams

- What are *Voronoi Diagrams*?
- Informally, Voronoi Diagrams are proximity regions to a set of objects. We collect all the points in a set for whom, some particulat proximity query gives the same result.
- Mostly we will be talking of *nearest point Voronoi Diagram*.

Voronoid Diagram

Example (Voronoi Diagram in Plane)



Proximity Queries

- A dataset S of n points called *sites* in $\Re^2$.
  Let $S = \{s_1, s_2, \ldots, s_n\}$.

Proximity Queries

- A dataset S of n points called *sites* in $\Re^2$.
  Let $S = \{s_1, s_2, \ldots, s_n\}$.
- (Query 1) For any given *query point* find the closest site to q.

Proximity Queries

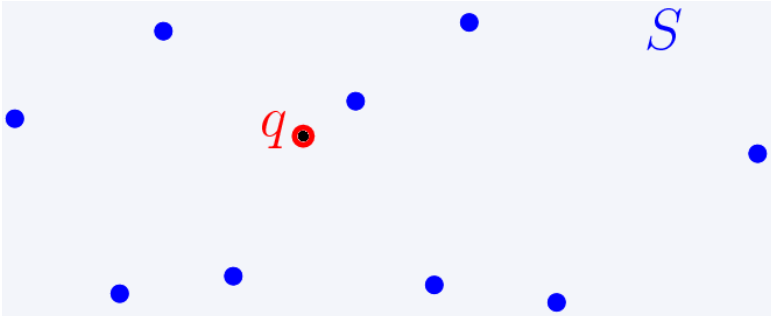
- A dataset S of n points called *sites* in $\Re^2$.
  Let $S = \{s_1, s_2, \ldots, s_n\}$.
- (Query 1) For any given *query point* find the closest site to q.
- (Query 2) Find the *proximity region* of a site.

Proximity Queries

- A dataset S of n points called *sites* in $\Re^2$.
  Let $S = \{s_1, s_2, \ldots, s_n\}$.
- (Query 1) For any given *query point* find the closest site to q.
- (Query 2) Find the *proximity region* of a site.
- (Query 3) Find the *proximity regions* of all sites.

Proximity Query — Illustration

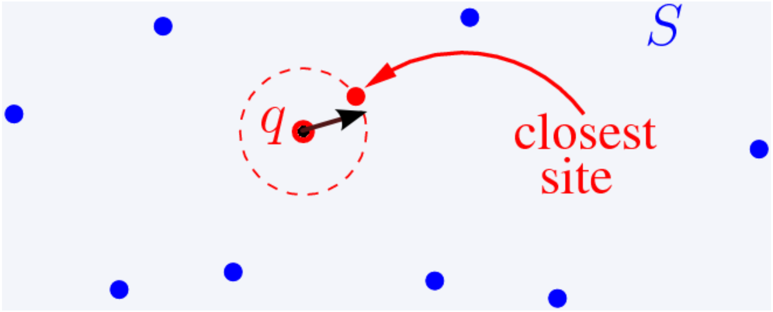
Example (Sites and Query Point)



What is nearest point to q?

Proximity Query — Illustration

Example (Nearest Site to Query Point)



Also, circle of shortest radius is empty of other sites.

Proximity Query — Easy Solution

- Naturally the solution can be found in linear time.
  Just calculate the distances from all sites and choose the closest site.

Proximity Query — Easy Solution

- Naturally the solution can be found in linear time.
  Just calculate the distances from all sites and choose the closest site.
- However, if you have to do it repeatedly, say m times, you can only do it in $O(nm)$-time—quadratic time by this method.

Proximity Query — Easy Solution

- Naturally the solution can be found in linear time.
  Just calculate the distances from all sites and choose the closest site.
- However, if you have to do it repeatedly, say m times, you can only do it in $O(nm)$-time—quadratic time by this method.
- Or, if you have to find the *proximity regions* themselves, the easy algorithm does not work. There are ∞ points.

What are Voronoi Diagrams (very formally)

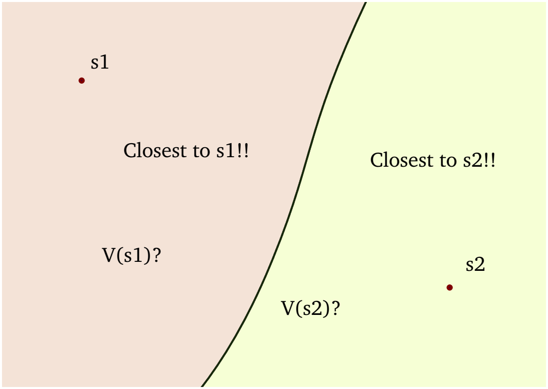
Definition

- Let S be a set of n distinct points, s_i, $\forall i \in n$, called *sites* in the plane
- The Voronoi diagram of S is the subdivision of the plane into n cells, $V(s_i)$, one for each site s_i,
- A point q lies in $V(s_i)$ iff $||q - s_i|| < ||q - s_j||$, for each $s_j \in S, i \neq j$

Simply put, $V(s_i)$ is the set of points whose nearest point is s_i.

Computing Voronoi Diagram

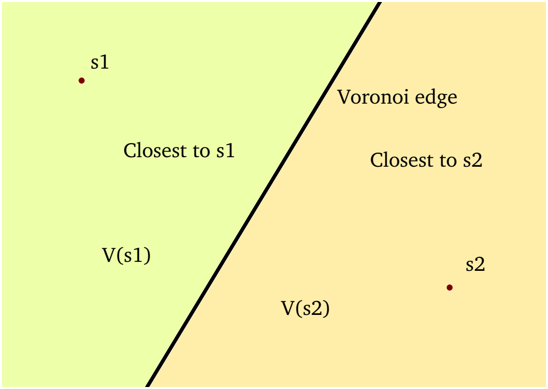
Example (What is Voronoi Diagram of 2 points)



Partition the plane in two sets. Points nearer to s_1 and nearer to s_2

Computing Voronoi Diagram

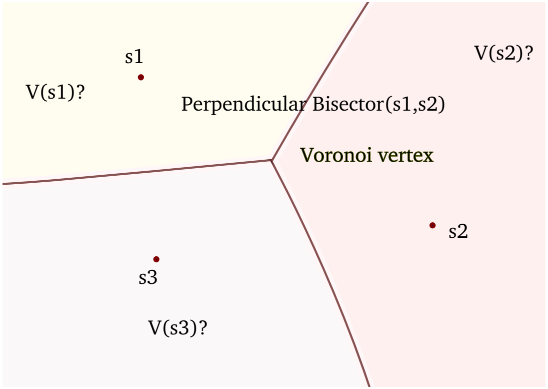
Example (Voronoi Diagram of 2 points)



It's perpendicular bisector

Computing Voronoi Diagram of 3 points

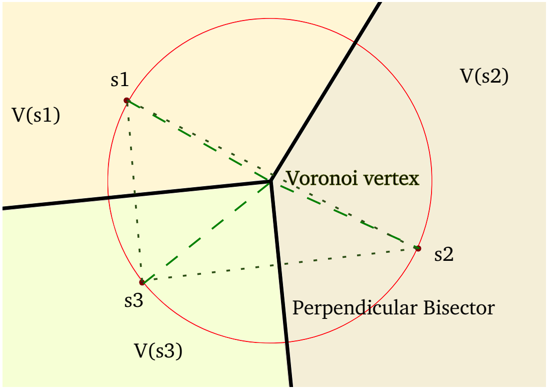
Example (What is Voronoi Diagram of 3 points)



Partition the plane in three sets. Points nearer to s_1, s_2 and s_3

Properties of Voronoi Diagram

Example (Voronoi Diagram of 3 points)



Bisectors intersect at circumcentre

Properties

- Voronoi edges are in part perpendicular bisectors.

Properties

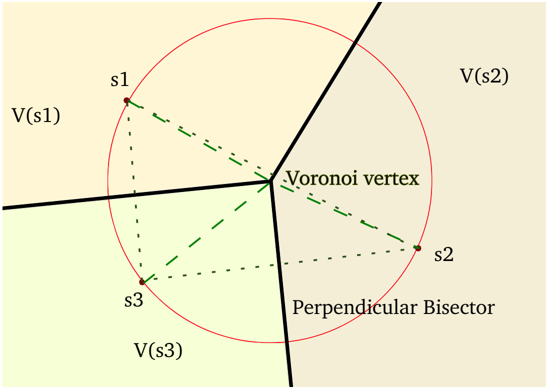
- Voronoi edges are in part perpendicular bisectors.
- Voronoi vertices are equidistant to three points (circumcentre).

Properties

- Voronoi edges are in part perpendicular bisectors.
- Voronoi vertices are equidistant to three points (circumcentre).
- Not immediately obvious, the circumcircle must be empty (Let us see this again).

Properties of Voronoi Diagram

Example (There can not be a point nearer than the three points)



Circumcircle is empty and voronoi vertex in centre of circumcircle.
This motivates an algorithm.

Straight-forward Algorithm

Frontal Attack to the Problem

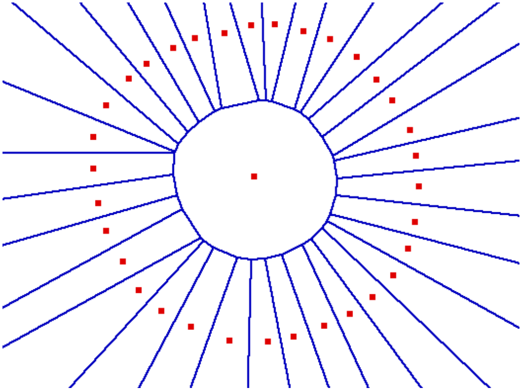
- Compute $\binom{n}{3}$ circumcentres
- Remove non-empty circumcircles
- Intelligently join circumcentres by bisectors (whenever two points are common)
- Voila, you get the Voronoi Diagram

Bad algorithm, $O(n^4)$ time complexity

Space Complexity of Voronoi Diagrams

Before giving good algorithms, we ask what is the size of output.
Why?
Because if output is big we do not hope to improve.

Bad Case for Space Complexity



A cell so big that it has $n - 1$ edges.

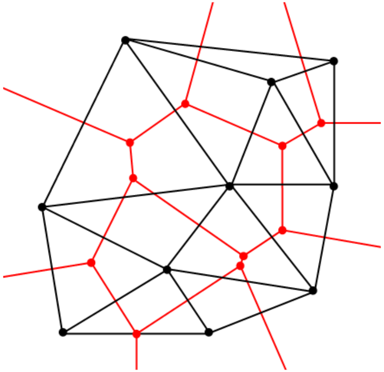
Size of Voronoi Diagram is Linear

However, we can prove that

- Number of Voronoi Vertices is $O(n)$
- Number of Voronoi Edges is $O(n)$
- So, Size of Voronoi Diagrams is $O(n)$

Proof of Space Complexity

We look at Dual of Voronoi Diagram (Delaunay Triangulation)



Voronoi vertices are Delaunay faces (f), Voronoi edges are Delaunay edges (e)

We need to bound both f and e

Proof of Space Complexity

- Euler's relation for planar graph (Delaunay's) is $v + f = e + 2$

Proof of Space Complexity

- Euler's relation for planar graph (Delaunay's) is $v + f = e + 2$
- $e \leq 3v - 6$, again from Euler,

Proof of Space Complexity

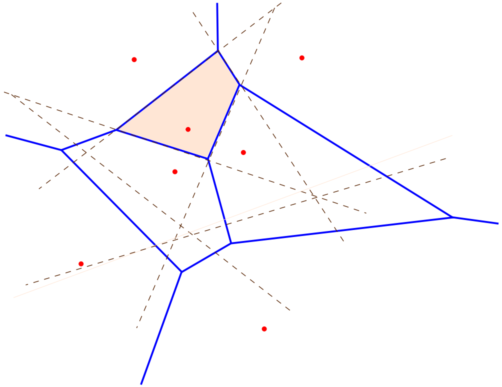
- Euler's relation for planar graph (Delaunay's) is $v + f = e + 2$
- $e \leq 3v - 6$, again from Euler,
- and therefore, $f \leq 2v - 4$

Proof of Space Complexity

- Euler's relation for planar graph (Delaunay's) is $v + f = e + 2$
- $e \leq 3v - 6$, again from Euler,
- and therefore, $f \leq 2v - 4$
- Since there is an implicit face at ∞, which corresponds to no voronoi vertex, number of voronoi vertices $f \leq 2v - 5$.

Another Algorithm

Each Voronoi cell is intersection of $n - 1$ half planes



We need to compute intersections efficiently

Another Algorithm—Analysis

- Each $V(s_i)$ is intersection of $n - 1$ half planes

Another Algorithm—Analysis

- Each $V(s_i)$ is intersection of $n-1$ half planes
- Intersection of n halfplanes can be computed in $O(n \log n)$ time

Another Algorithm—Analysis

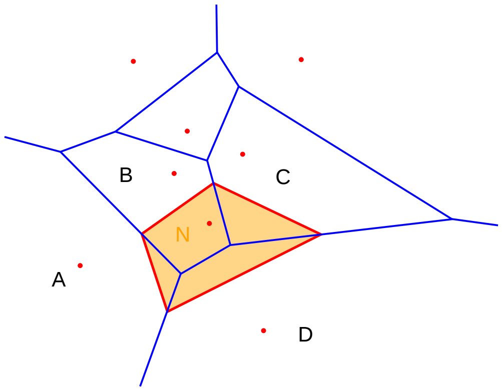
- Each $V(s_i)$ is intersection of $n-1$ half planes
- Intersection of n halfplanes can be computed in $O(n \log n)$ time
- Voronoi diagram of S can be computed in $O(n^2 \log n)$ time by computing the Voronoi cells one by one.

Another Algorithm—Analysis

- Each $V(s_i)$ is intersection of $n-1$ half planes
- Intersection of n halfplanes can be computed in $O(n \log n)$ time
- Voronoi diagram of S can be computed in $O(n^2 \log n)$ time by computing the Voronoi cells one by one.
- A large improvement over previous method (from n^4)

Yet Another Algorithm (Incremental)

Each point is inserted in Voronoi Diagram one by one



Yet Another Algorithm (Analysis)

- Site s_{i+1} is located in Voronoi Diagram of i sites (by ray shooting?)

Yet Another Algorithm (Analysis)

- Site s_{i+1} is located in Voronoi Diagram of i sites (by ray shooting?)
- Its boundary is calculated

Yet Another Algorithm (Analysis)

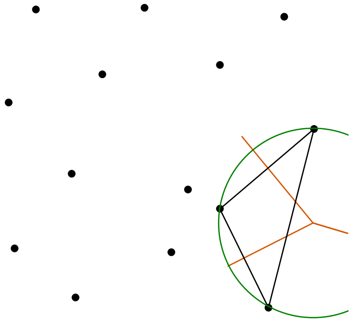
- Site s_{i+1} is located in Voronoi Diagram of i sites (by ray shooting?)
- Its boundary is calculated
- Voronoi diagram of S can be computed in $O(n^2)$ as $O(i)$ edges might be inserted or deleted in each step

Yet Another Algorithm (Analysis)

- Site s_{i+1} is located in Voronoi Diagram of i sites (by ray shooting?)
- Its boundary is calculated
- Voronoi diagram of S can be computed in $O(n^2)$ as $O(i)$ edges might be inserted or deleted in each step
- Further improvement over previous method

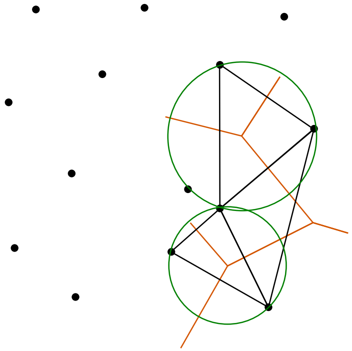
One more Incremental Algorithm-I

Each Voronoi vertex is inserted in Voronoi Diagram one by one



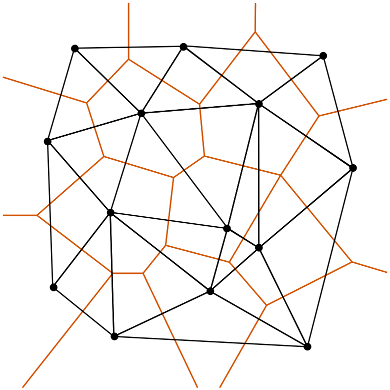
We start with any hull edge and compute *the* empty circle

One more Incremental Algorithm-II



Compute adjacent empty circles

One more Incremental Algorithm-III



Final Voronoi Diagram

Complexity Analysis of Incremental Algorithm

This algorithm also computes Voronoi Diagram in $O(n^2)$

Even though the discussed algorithms are non-aptimal, they are useful for Voronoi Diagram generalisations where the structures are complex and cannot be calculated using usual methods

Can we Compute Voronoi Diagram Faster?

- Clearly no method is optimal.
- Can we do better?

Three Popular Algorithms

We know of three good algorithms.

- Fortune's Sweep Line (Beach Line) Method
- Divide and Conquer
- Reduction to Convex Hull in $\Re^3$

Reduction to Convex Hull in $\Re^3$

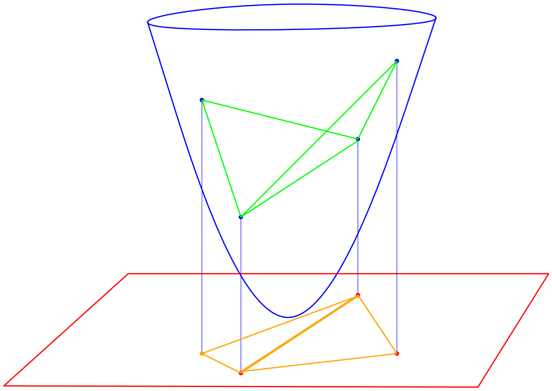
Transform points in plane to space

$(x, y) \rightarrow (x, y, x^2 + y^2)$

Basically we project the plane to the surface of a paraboloid
The lower convex hull is Delaunay Triangulation, the dual of Voronoi Diagram

Projection to Paraboloid

Reduction to Convex Hull



Lower Convex Hull is Delaunay Triangulation.

Fortune's Algorithm

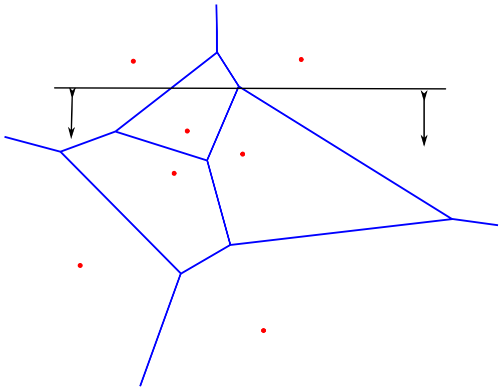
Assumptions

General position assumption: No four sites are co-circular.

Figures sources:
http://www.ams.org/featurecolumn/archive/voronoi.html

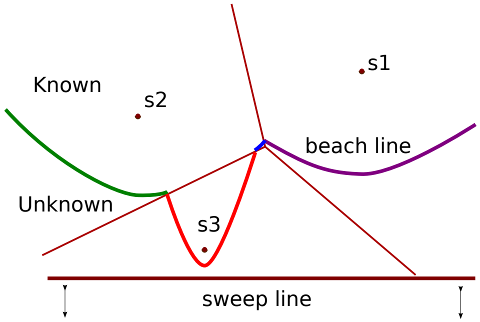
Why Usual Sweep Line Methods Fail?

The problem with usual sweeping methods is that the cells may start even before the site is encountered.



Tentative Idea

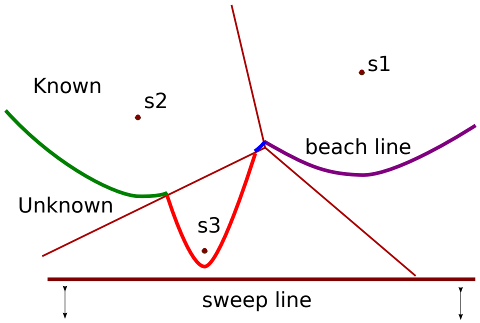
Fortune's algorithm is sweep line method [6]



Region inside beach line does not depend on anything below the sweep line. What is it?

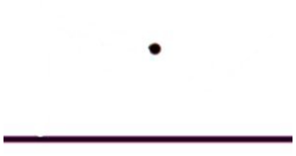
Tentative Idea

Fortune's algorithm is sweep line method [6]



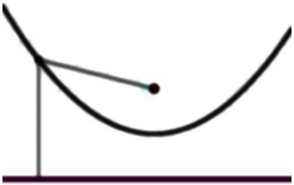
Beach line consists of sequence of parabolas. Why?

What is Beach line-0



We want to find where Voronoi Diagram will not change (once computed).
What are the points in plane always nearer to the point than anything below the sweep line?

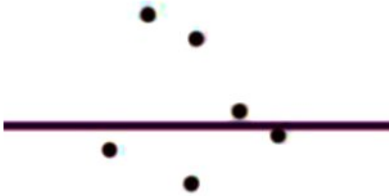
What is Beach line-I



It is parabola.
We know equidistant points from a site and a line is a parabola.
Anything in the parabola is nearer to the point always.

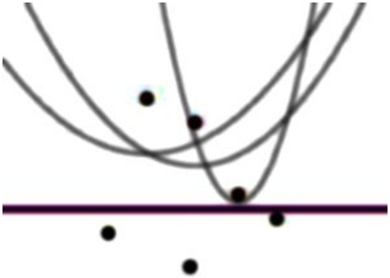
What is Beach line-II

What if we have more points?



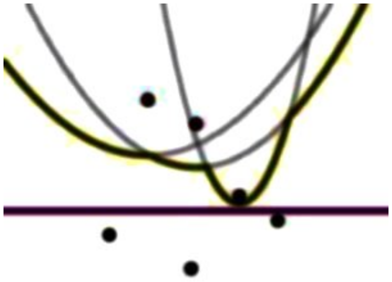
Suppose we have got six sites

What is Beach line-III



Three sites are already seen by sweep line. Another three are yet to be seen.

What is Beach line-IV



Beach line will be the envelope of the three parabolae

Important Property of Beach Line

We can order the parabolic segments from left to right (or right to left)

Beach line intersects any vertical line only once

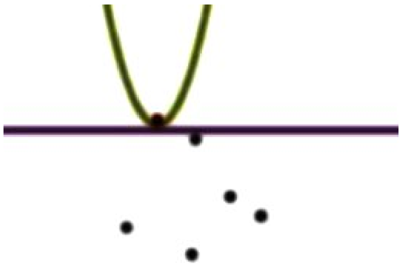
Two types of Events

Sweep line can have two types of events

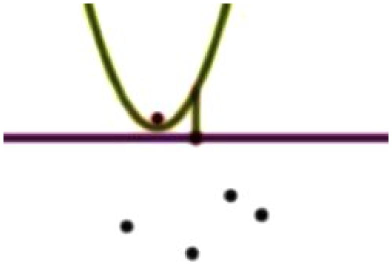
- Site Event (Voronoi Edges)
- Circle Event (Voronoi Vertices)

The algorithm basically keeps track of the beach line, while computing Voronoi Diagram inside it.

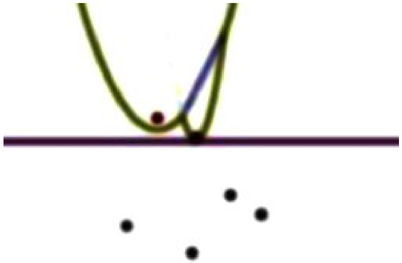
Initial Situation



A site is encountered

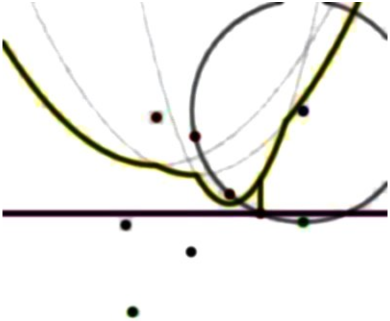


Voronoi Edge Enters



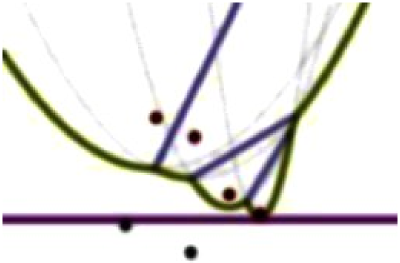
That means we have a Voronoi edge whenever two beach segments intersect

Voronoi Vertex Enters



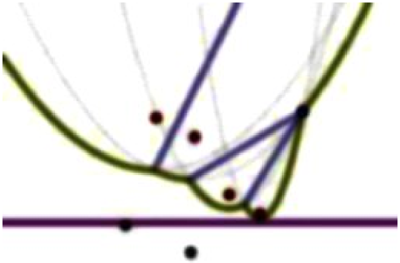
There is a Voronoi vertex whenever two Voronoi edges intersect
We need to add an event (circle event) where three neighbouring beach segments intersect

Circle Event-I



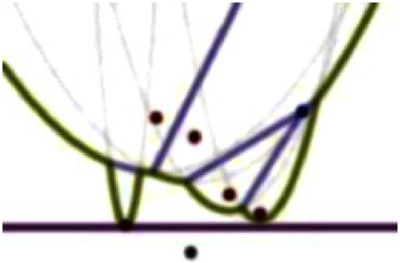
Before circle event

Circle Event-II



At the circle event

Circle Event-III

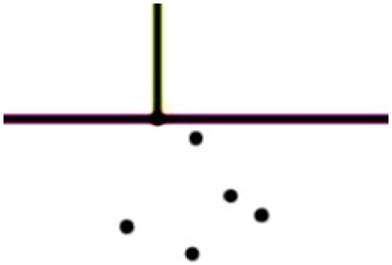


After circle event

Execution of the Fortune's Algorithm

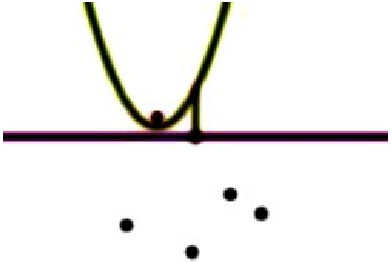
Let us see how the Fortune's algorithm calculates the Voronoi Diagram step by step

Algorithm—Step 1



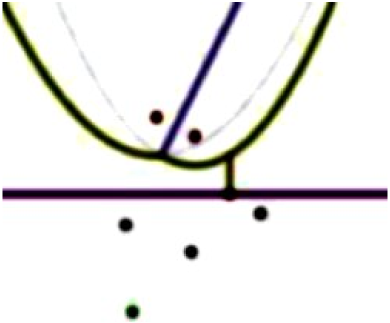
First site and a strange beach line

Algorithm—Step 2



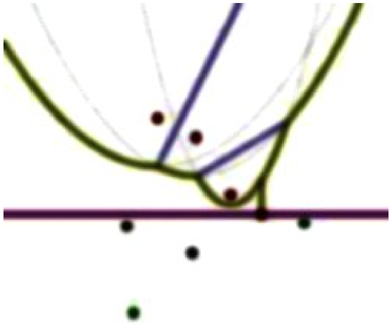
Second site

Algorithm—Step 3



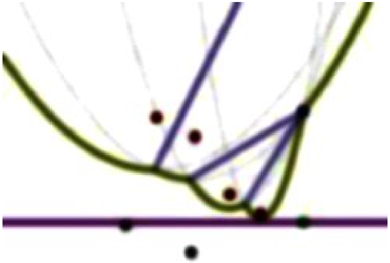
Third site (and an ineffective circle event)

Algorithm—Step 4



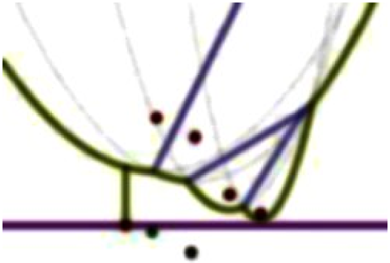
Fourth site (and an effective circle event)

Algorithm—Step 5



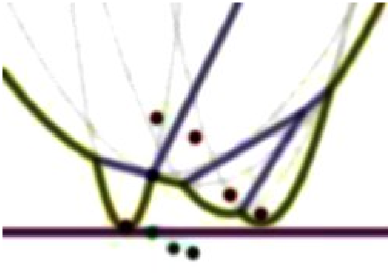
Circle event

Algorithm—Step 6



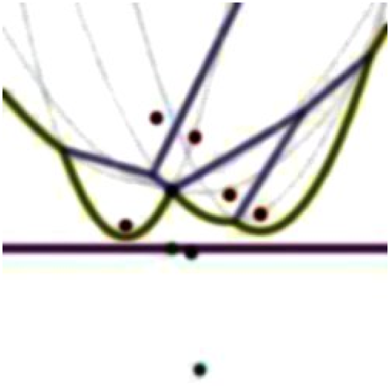
Fifth site and a circle event

Algorithm—Step 7



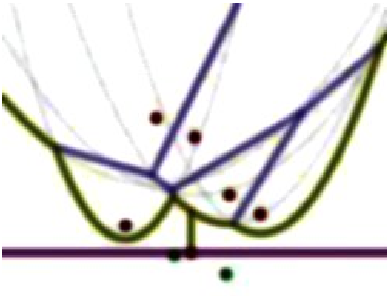
A circle event and another circle event

Algorithm—Step 8



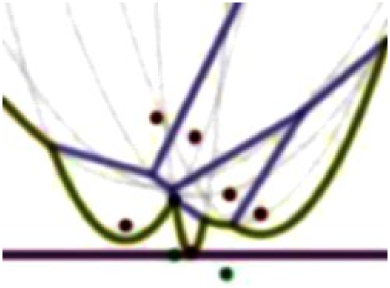
Another of the same

Algorithm—Step 9



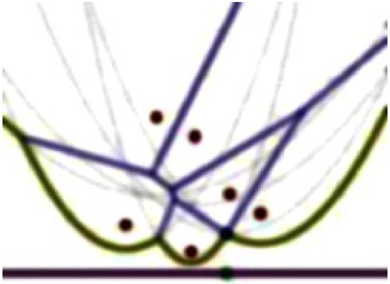
Last site and two circle event

Algorithm—Step 10



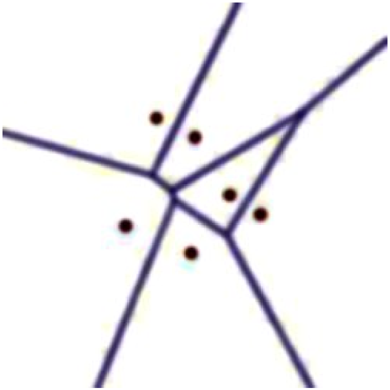
Last but one circle event

Algorithm—Step 11



Last circle event

Algorithm—Step 12



Final Output

Analysis of the Fortune's Algorithm

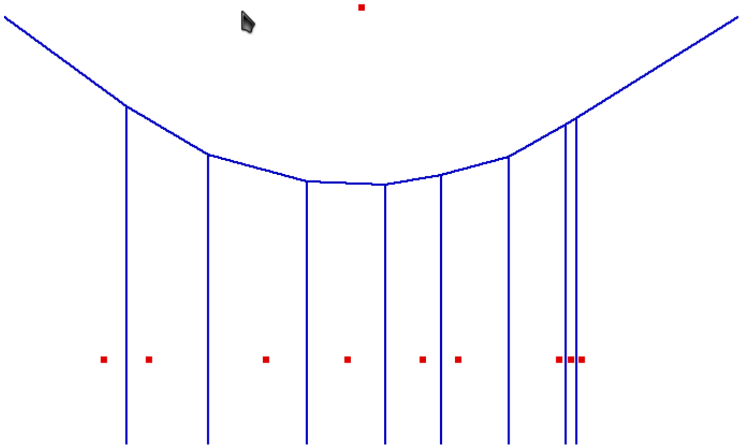
Fortune's algorithm being an example of typical sweep line technique is $O(n \log n)$

Optimal because sorting problem can be reduced to construction of Voronoi Diagrams

Reduction of Sorting Problem to Voronoi Diagram

To sort $x_1, x_2, \ldots, x_n$
We find Voronoi Diagram of points
$(x_1, 0), (x_2, 0), \ldots (x_n, 0)$ and $(0, \infty) = s_\infty$
Either neighbourhood relation or the cell $V(s_\infty)$ gives the required sorted order

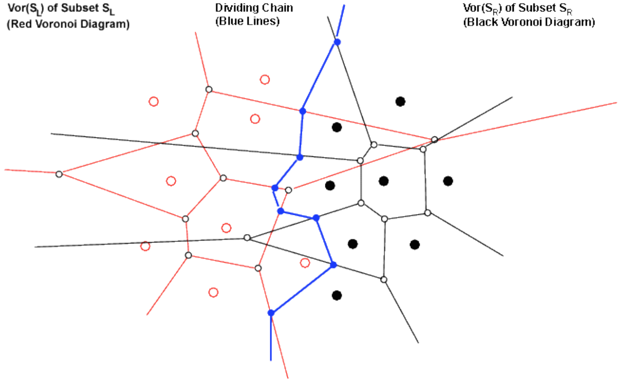
Sorting using Voronoi Diagrams



Divide and Conquer

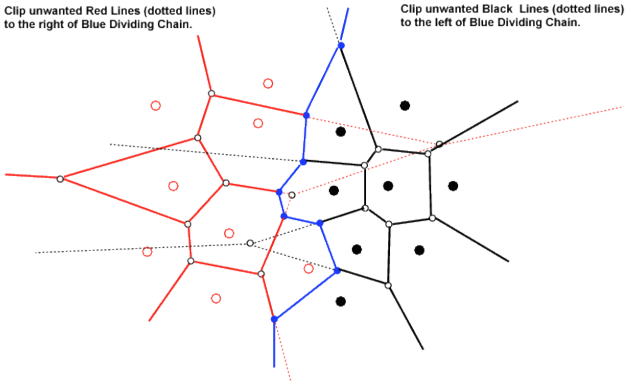
Another algorithm that gives Voronoi Diagram in optimal time and space

Divide


$\mathrm{Vor}(S_L)$ of Subset S_L
(Red Voronoi Diagram)
Dividing Chain
(Blue Lines)
$\mathrm{Vor}(S_R)$ of Subset S_R
(Black Voronoi Diagram)

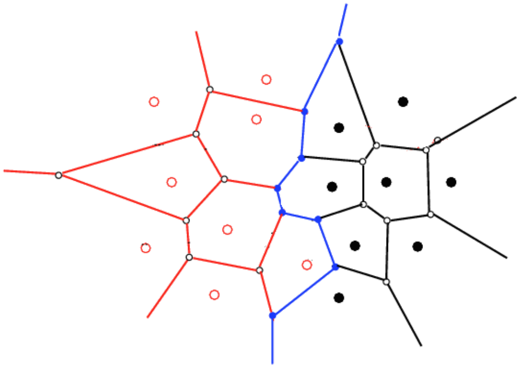
Partition points equally. Solve two subproblems.

Conquer


Clip unwanted Red Lines (dotted lines) to the right of Blue Dividing Chain.
Clip unwanted Black Lines (dotted lines) to the left of Blue Dividing Chain.

Sew the two jumbled pieces

Solution



Merging of Voronoi Diagram complete!!

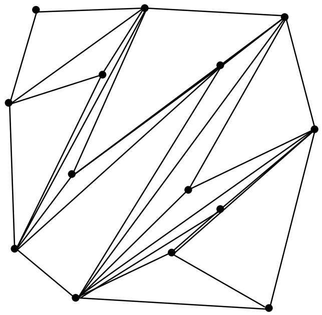
Delaunay Triangulations

What are Delaunay Triangulation?

- Given a set of points sometimes it is needed to join them in triangles

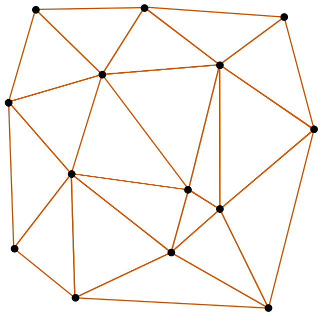
Bad triangulation

Example (Skinny Triangles)



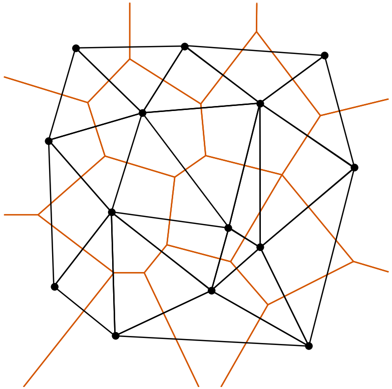
Good triangulation

Example (Delaunay Triangulation)



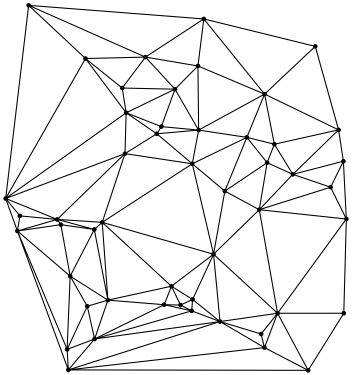
Delaunay triangulation as Dual of Voronoi Diagrams

Example (Delaunay Triangles are Voronoi Vertices)



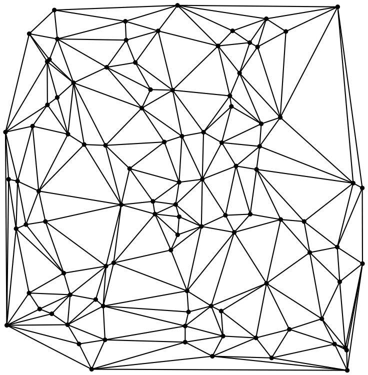
Another Example to Convince

Example (Delaunay Triangulation of 50 points)



Yet Another Example to Convince

Example (Delaunay Triangulation of 100 points)

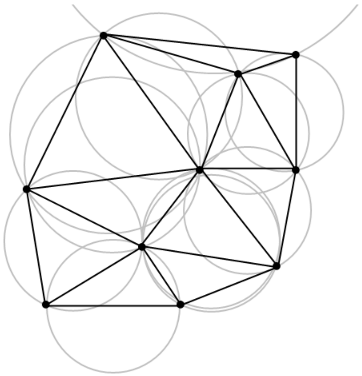


What are we looking for?

- Triangles should not be very stretched (skinny triangles)
- Maximize the minimum angle of all the angles of the triangles in the triangulation
- Circumcircles of all the triangles in the net are empty

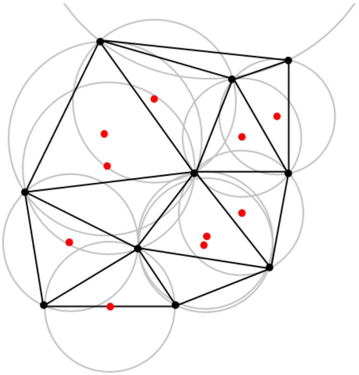
Delaunay Triangulation Properties-I

Example (Delaunay Triangles have Empty Circumcircles)



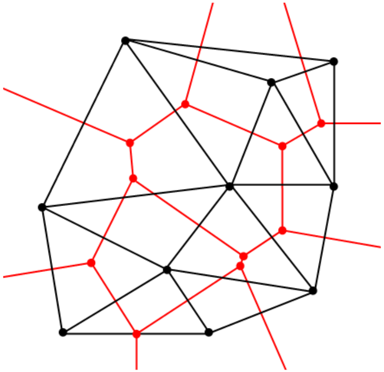
Delaunay Triangulation Properties-II

Example (Empty Circumcircles centred at Voronoi Vertices)



Delaunay Triangulation Properties-II

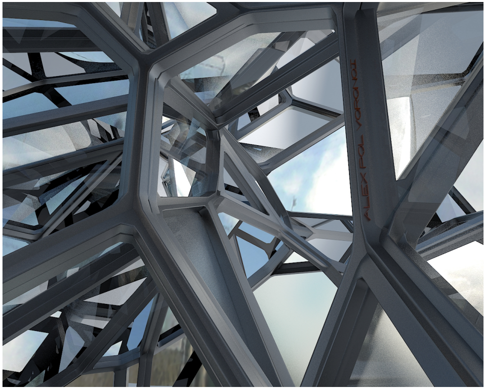
Example (Empty Circumcircles—Dual of Voronoi Diagrams)



Generalisations

Higher Dimension Voronoi Diagram

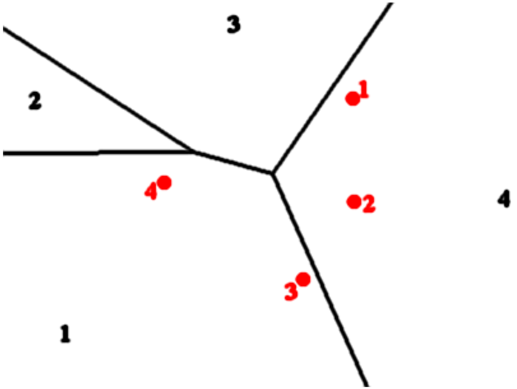
Example (Voronoi Region 3D point set)



Voronoi Diagram in space

Farthest Point Voronoi Diagram

Example (Voronoi regions for farthest point queries)



Computing Farthest Point Voronoi Diagram

Farthest Point Voronoi diagrams only depends on points on Convex Hull .

This makes the task of computing the Farthest Point Voronoi Diagram easier and gives rise to $O(n \log n)$ algorithm.

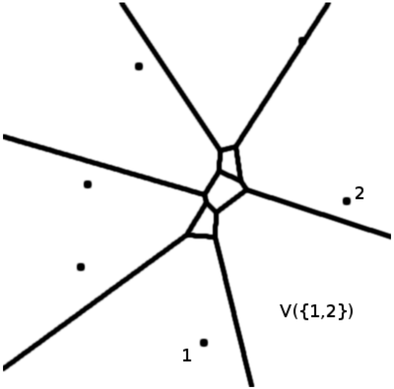
Applications of Farthest Point Voronoi Diagram

Used for Min-Max optimisation problems

Gives you minimum enclosing circle directly. Though linear time algorithms that do not use farthest point Voronoi Diagram are known.

Higher Order Voronoi Diagram

Example (Voronoi Region for k-nearest point set)



Second order Voronoi Diagram

Computing Higher Order Voronoi Diagram

Higher-order Voronoi diagrams can be generated recursively.

To generate the n^{th}-order Voronoi diagram from set S, start with the $(n-1)^{th}$-order diagram and replace each cell generated by $X = \{x_1, x_2, ..., x_{n-1}\}$ with a Voronoi diagram generated on the set $S - X$.

Collect all the regions of X in different cells.

Conclusion

Open Problems to Ponder

- Compute Voronoi diagram of a set of lines (or line segments) in three dimensions, conjectured to be near quadratic. (Same for Polyhedra is hard)
- What is the maximum number of combinatorial changes possible in a Euclidean Voronoi diagram of a set of n points each moving along a line at unit speed in two dimensions?
- Where to place a new site to maximize its Voronoi cell?

Summary

- We studied the concept of Voronoi Diagrams
- Next we saw how they can be computed
- We looked into their duals—Delaunay Triangulations
- Finally we also saw some of the generalisations

You may read

- Book by Okabe et al. [7], who lists more than 600 papers, and
- the surveys by Aurenhammer [1],
- Bernal [2], and
- Fortune [6] for a complete overview,
- Also, Chapters 5 and 6 of Preparata and Shamos [8], and
- Chapter 13 of Edelsbrunner [5]

References I

Franz Aurenhammer.
Voronoi diagrams - a survey of a fundamental geometric data structure.
ACM Comput. Surv., 23(3):345–405, 1991.

Javier Bernal.
Bibliographic notes on voronoi diagrams.
Technical report, Natl. Inst. of Standards and Tech., April 1993.
NIST Internal Report 5164.

Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf.
Computational Geometry: Algorithms and Applications.
Springer-Verlag, second edition, 2000.

References II

Rene Descartes.
Principia Philosophiae.
Ludovicus Elzevirius, Amsterdam, 1644.

Herbert Edelsbrunner.
Algorithms in Combinatorial Geometry.
Springer-Verlag, New York, 1987.

Steven Fortune.
Voronoi diagrams and Delaunay triangulations.
CRC Press, Inc., Boca Raton, FL, USA, 1997.

References III

Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu.
Spatial tessellations: Concepts and applications of Voronoi diagrams.
Probability and Statistics. Wiley, NYC, 2nd edition, 2000. 671 pages.

F. P. Preparata and M. I. Shamos.
Computational Geometry: An Introduction.
Springer-Verlag, 1985.

At Last . . .

Thank You

`shreesh@rkmvu.ac.in`
`sarvottamananda@gmail.com`