Voronoi Diagrams and their Applications

Maria Despoina Siampou (sdi1600151), George Michas (sdi1400109)

March 22, 2019

Contents

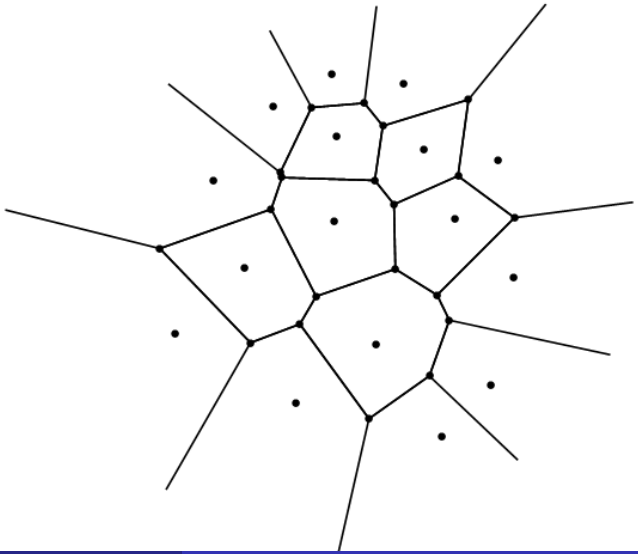
Contents

Voronoi Diagram - Introduction

A **Voronoi diagram** is a partitioning of a plane into regions based on distance to points in a specific subset of the plane.

That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other.

These regions are called **Voronoi cells**.



Voronoi Diagram - Introduction

Voronoi's Diagram Definition in R^2

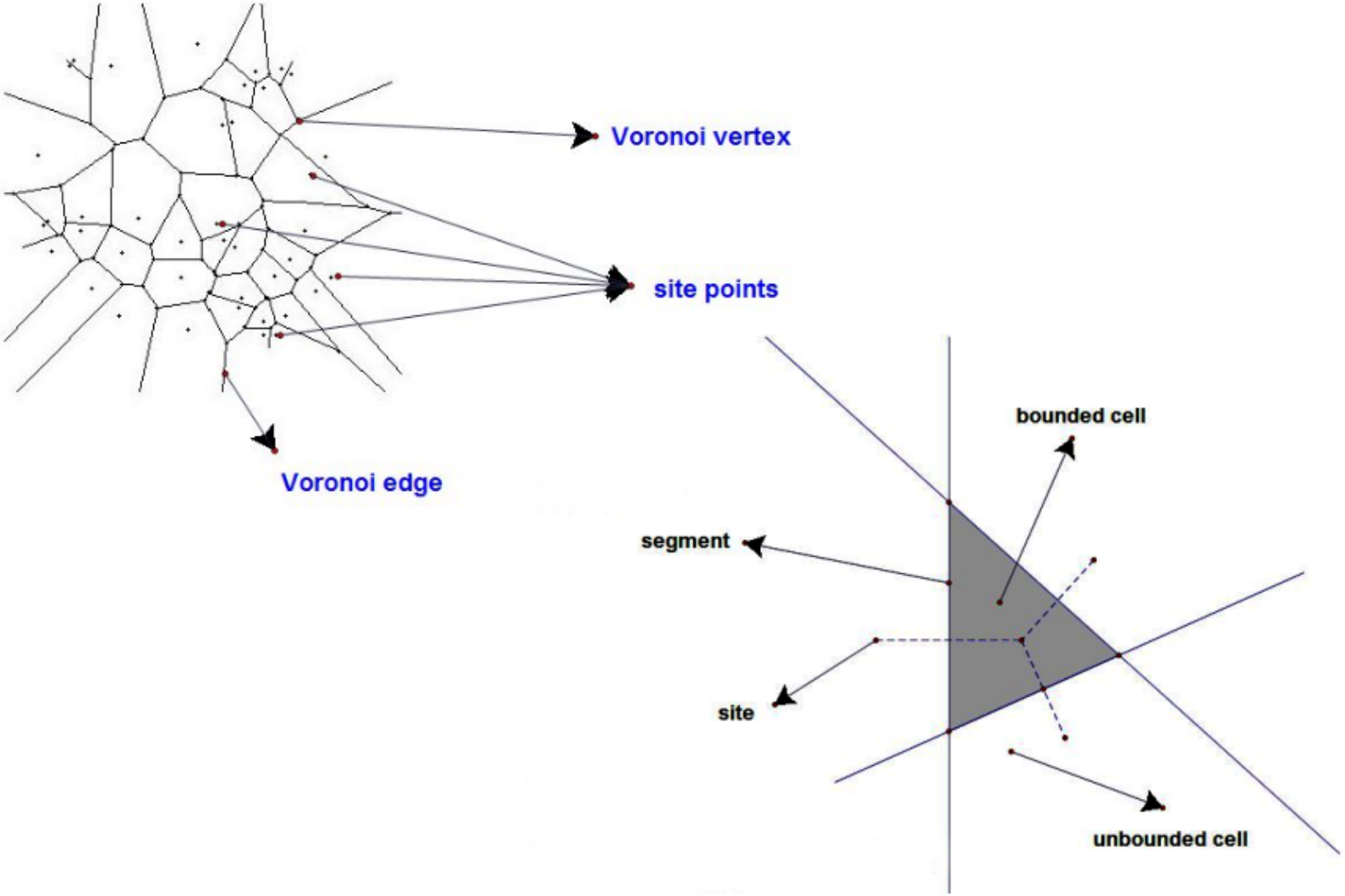
- Let $P = \{p_1, p_2, ..., p_n\} \in R^2$, where $n \geq 2$, $p_i \neq p_j$, $\forall i, j = \{1, 2, ..., n\}$ be the set of generator points called **Voronoi Seeds** or **Voronoi Sites** with corresponding vectors $\vec{x}$ for every $p_i \in P$.
- We define **Voronoi Region** or **Voronoi Cell** of p_i as :

$$V(p_i) = \{\vec{x} \mid ||\vec{x} - \vec{x_i}|| \leq ||\vec{x} - \vec{x_j}|| \, \forall \, j \, \exists \, i \neq j\}$$

where $||\vec{p}, \vec{q}|| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, $\vec{p} = (x_i, y_i), \vec{q} = (x_j, y_j)$

- **Voronoi Diagram** is the set V= $\bigcup_{i=1}^{n} V_{p_i}$

Voronoi Diagram

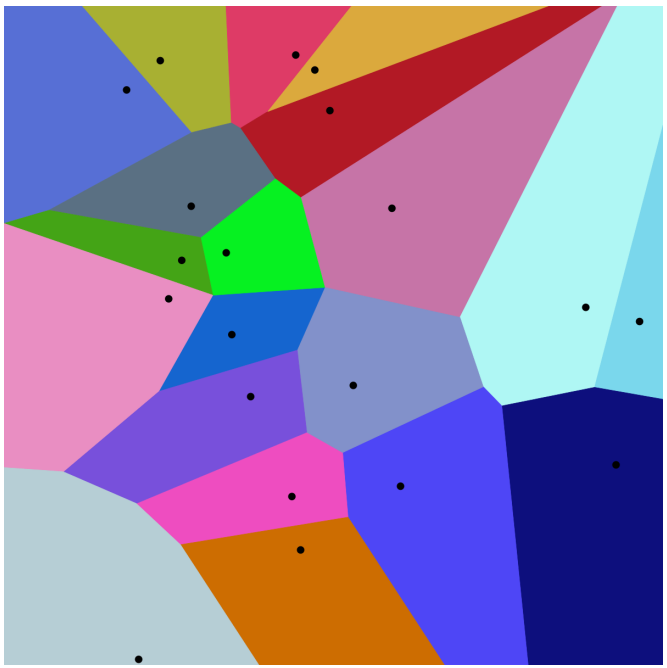


Illustration

The distance between points can be measured using the familiar **Euclidean** or **Manhattan** distance.

$$l_2 = d[(a_1, a_2), (b_1, b_2)] = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

$$l_2 = d[(a_1, a_2), (b_1, b_2)] = |a_1 - b_1| + |a_2 - b_2|$$



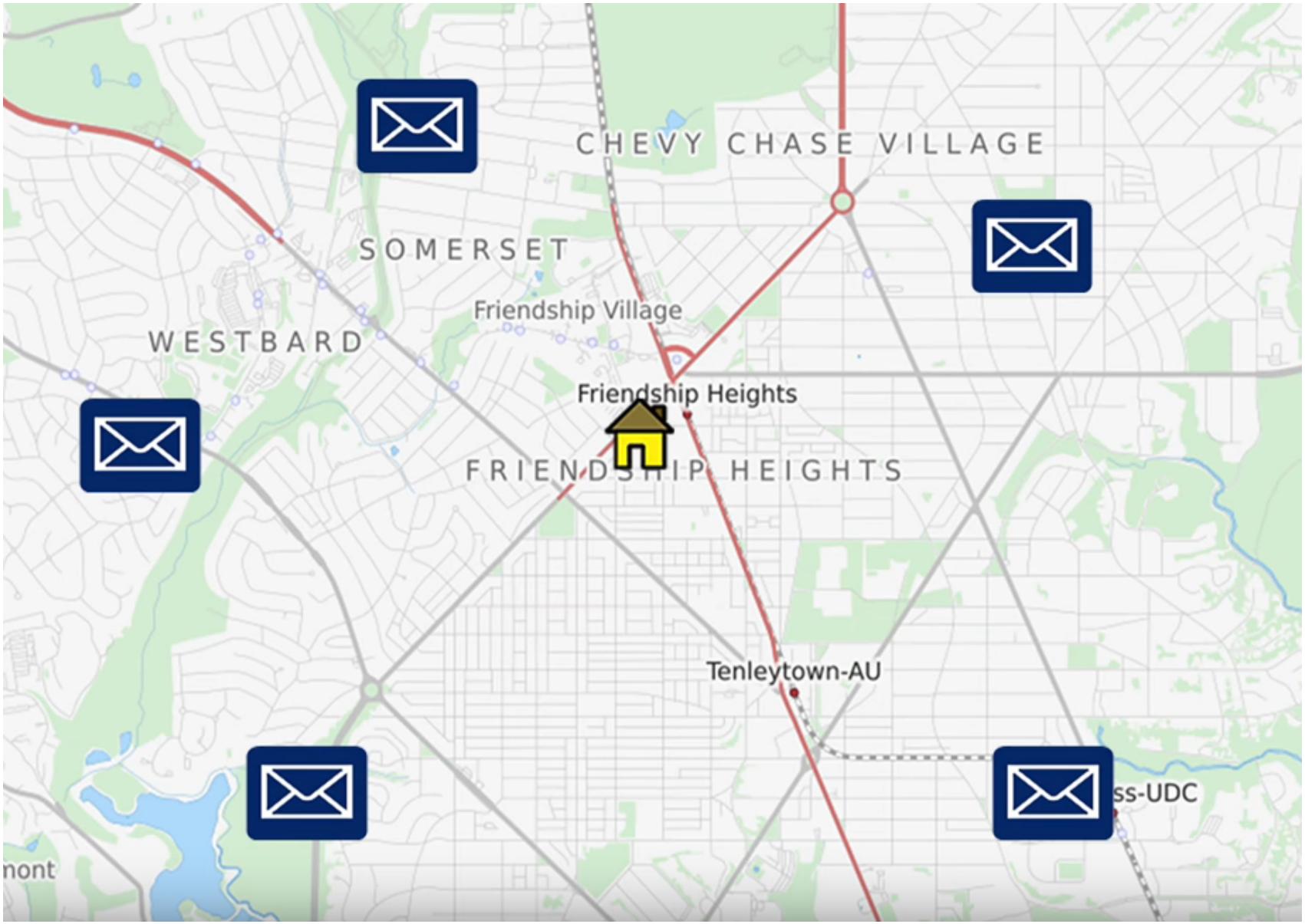
Contents

Post Office Problem

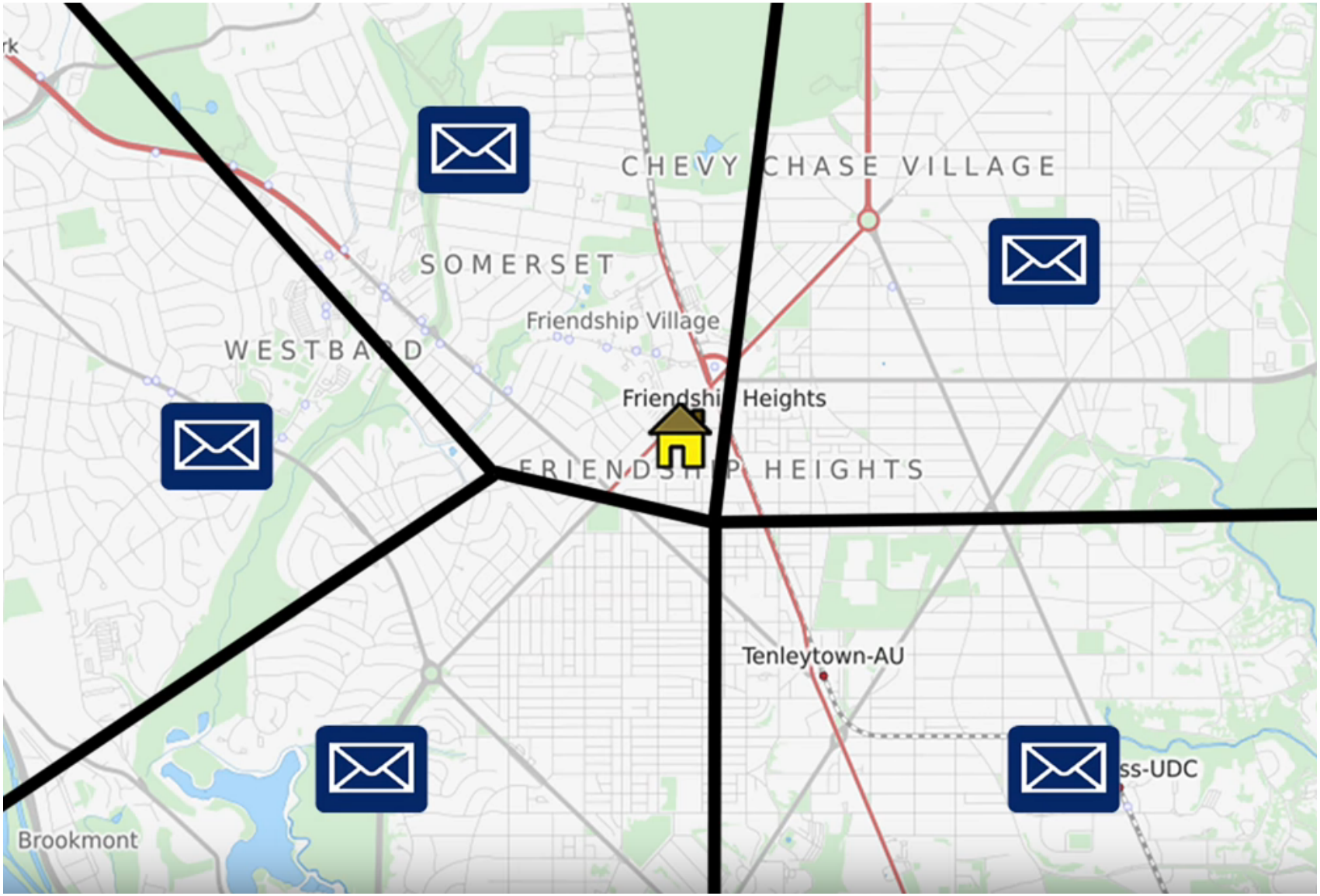
Consider a classic optimization problem called the **Post Office Problem**.

A new city has just built 10 post offices. With the aim of reducing the cost of delivering mail, how can we optimally assign each residence a post office from which to receive mail?

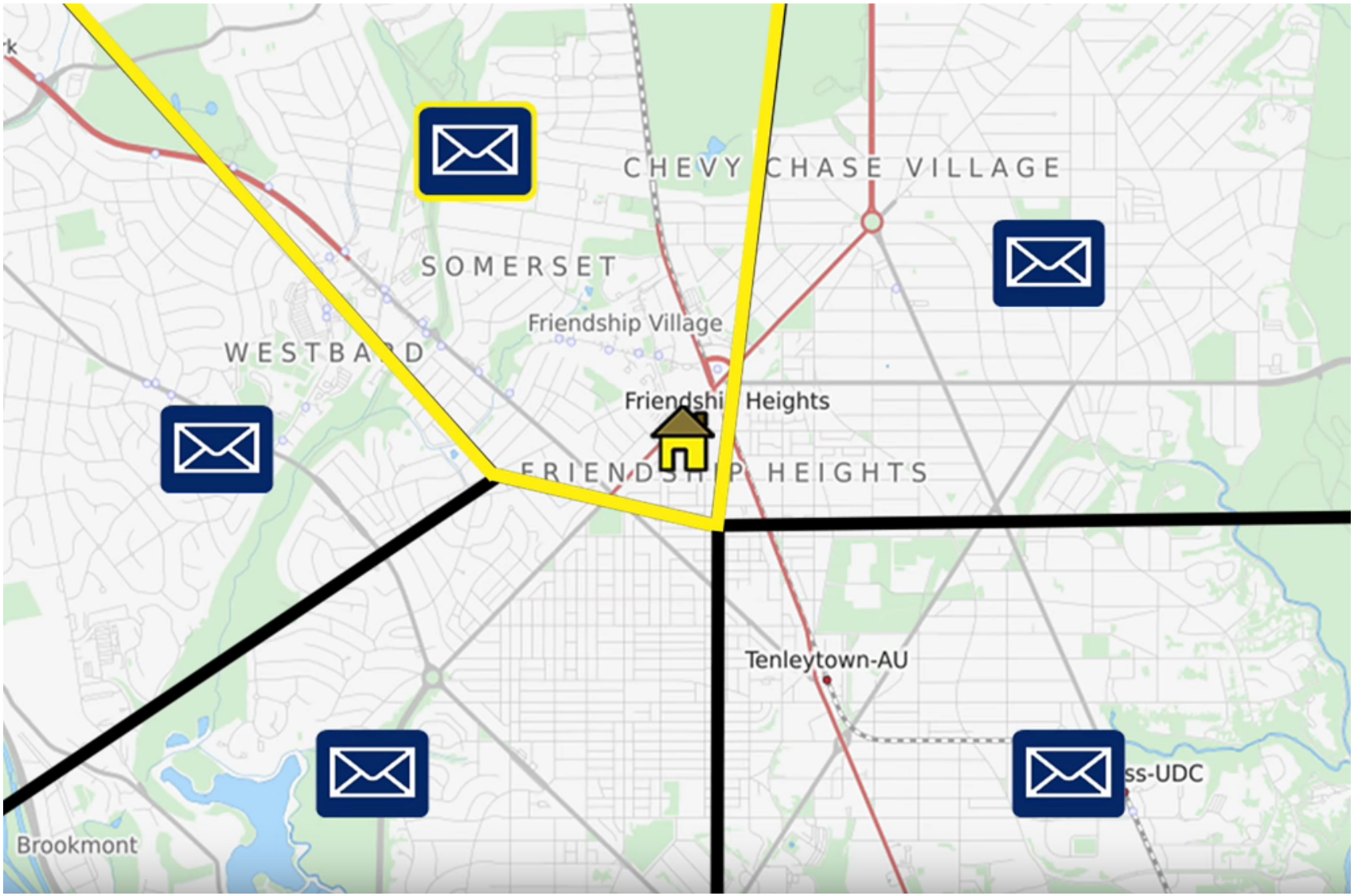
Post Office Problem



Post Office Problem

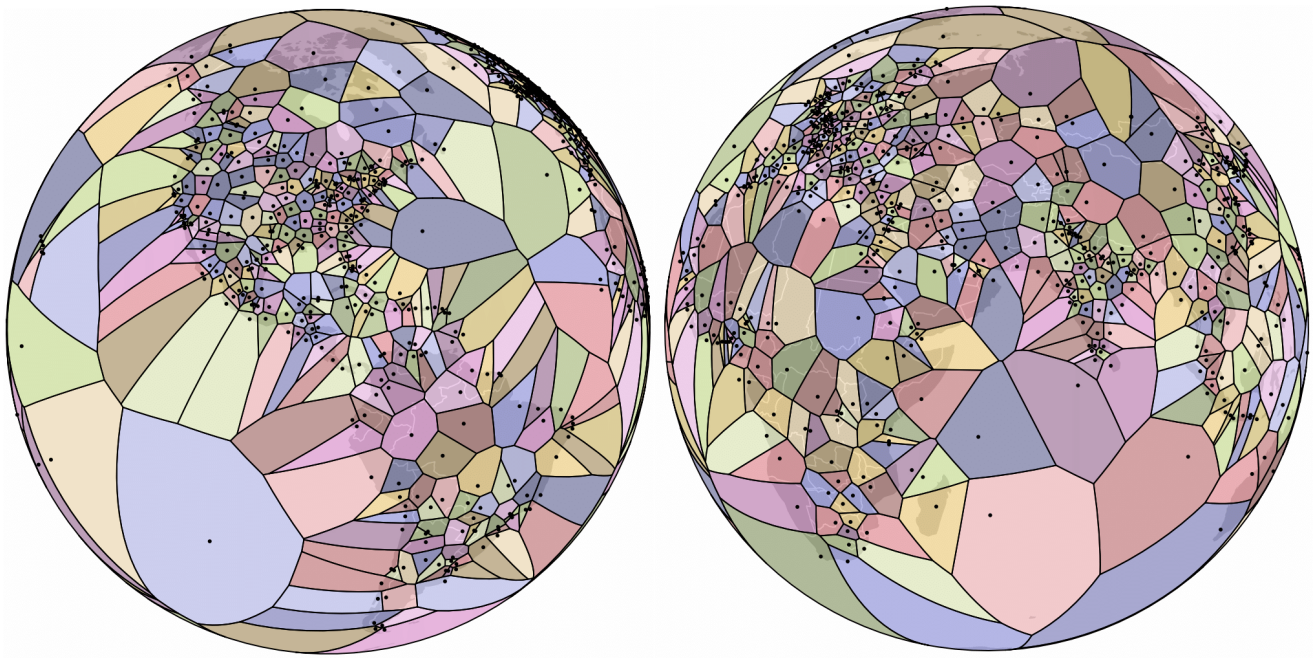


Post Office Problem

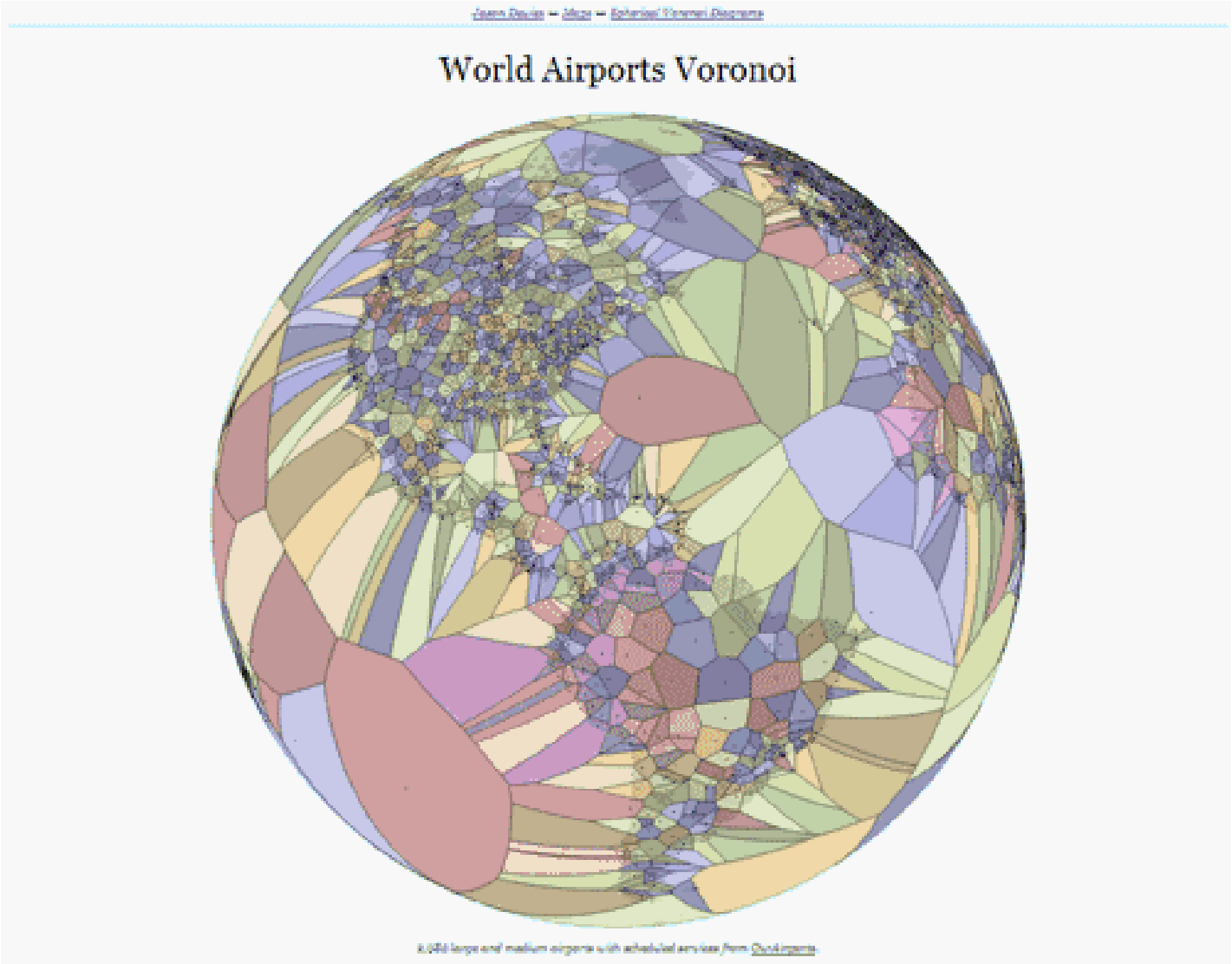


World Airports Voronoi

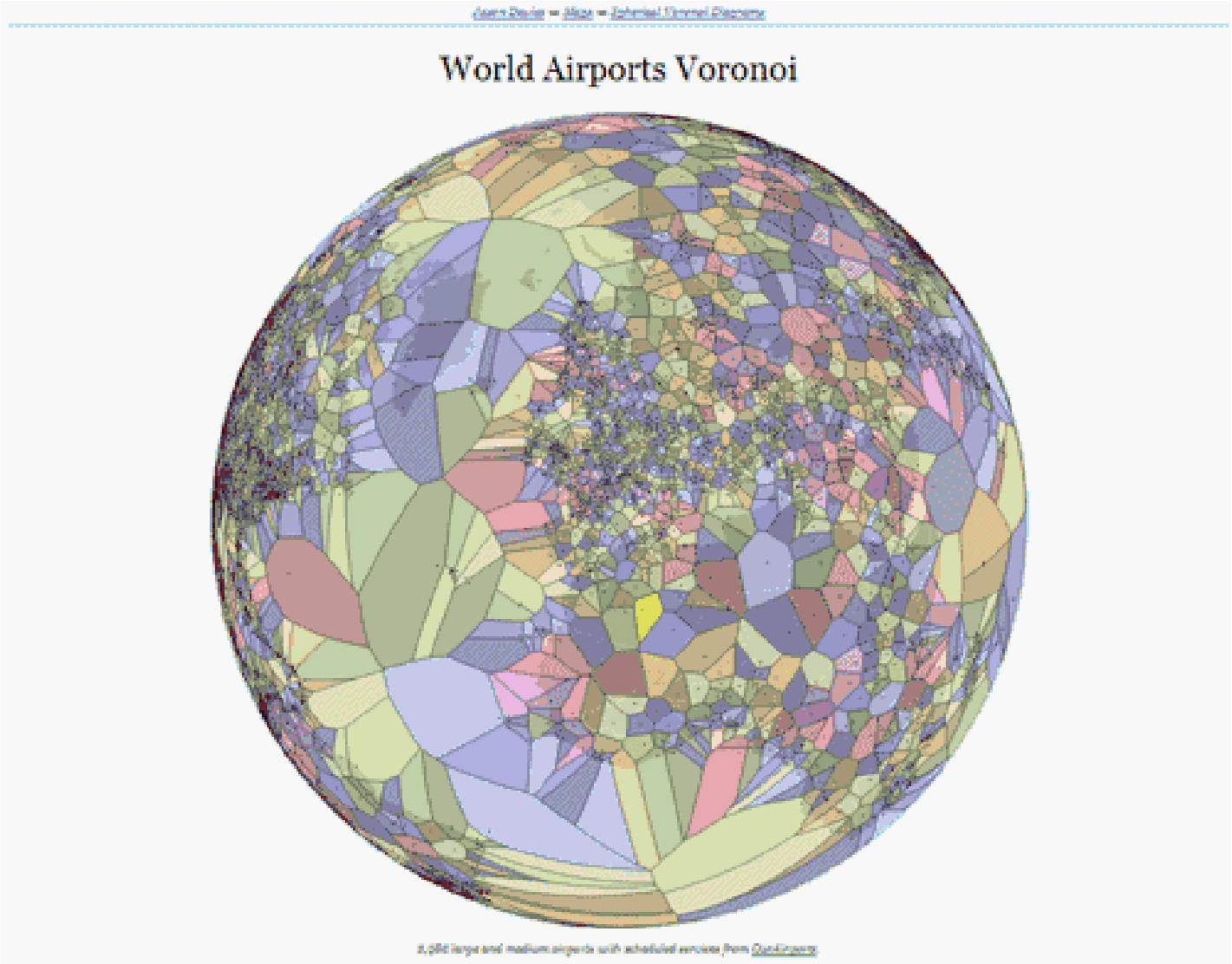
Each region is closer to a particular airport than any other. This partitioning of the sphere is called a **spherical Voronoi diagram**.



World Airports Voronoi



World Airports Voronoi



Uncovering the cause of cholera

London, September, 1854. A cholera outbreak has decimated Soho, killing 1/10 of the population.

John Snow created an ingenious map that dramatically showed the geographic spread of deaths in the outbreak.

This map supported Snow's hypothesis that the cholera deaths were associated with contaminated water, in this case, from the Broad Street Pump.

Snow's work helped develop the modern field of epidemiology, and this map has been called "The most famous 19th century disease map".

Uncovering the cause of cholera

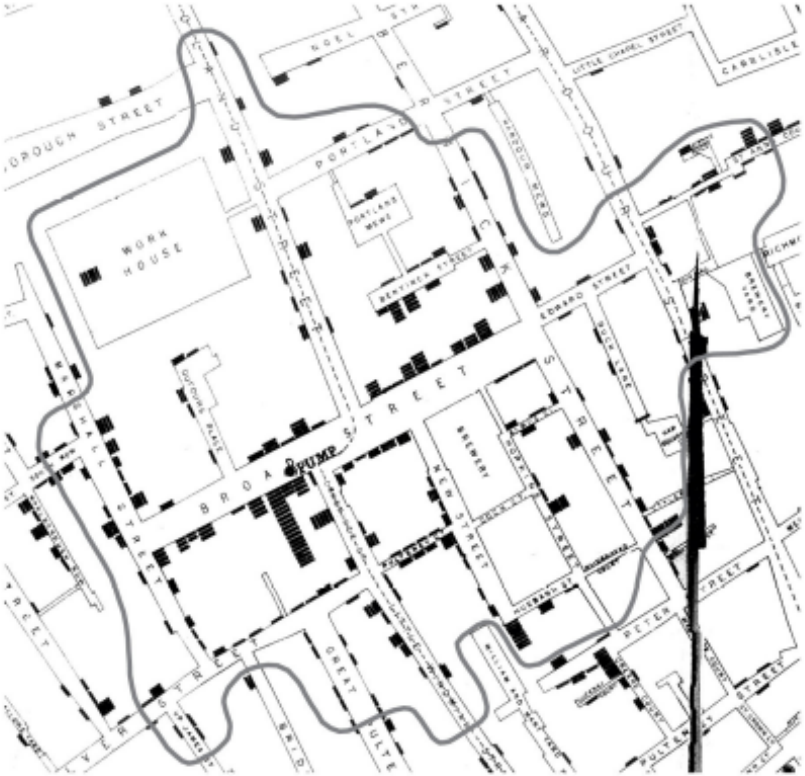


Figure: Each bar represents a death at an address. The curve marks points at equal distance from the Broad Street pump and another pump.

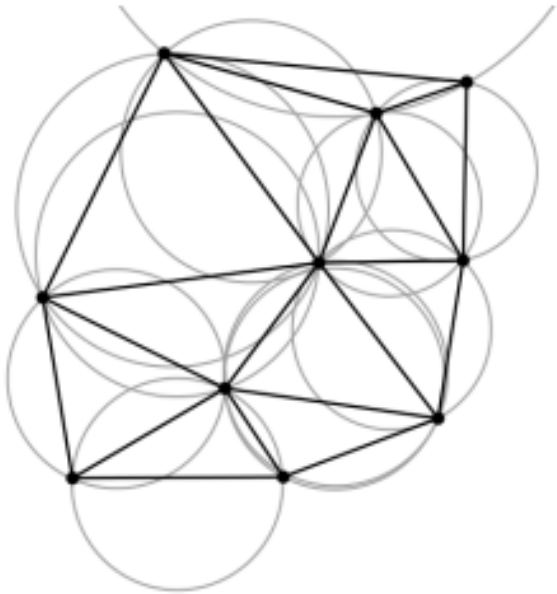
Contents

Delaunay Triangulation

Delaunay triangulation is a triangulation of a plane using a set of discrete points, so that no point is inside the circumcircle of any triangle.

Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation.

This can also be used in higher dimensions.



Relationship with the Voronoi diagram

Delaunay triangulation is a dual graph of Voronoi.

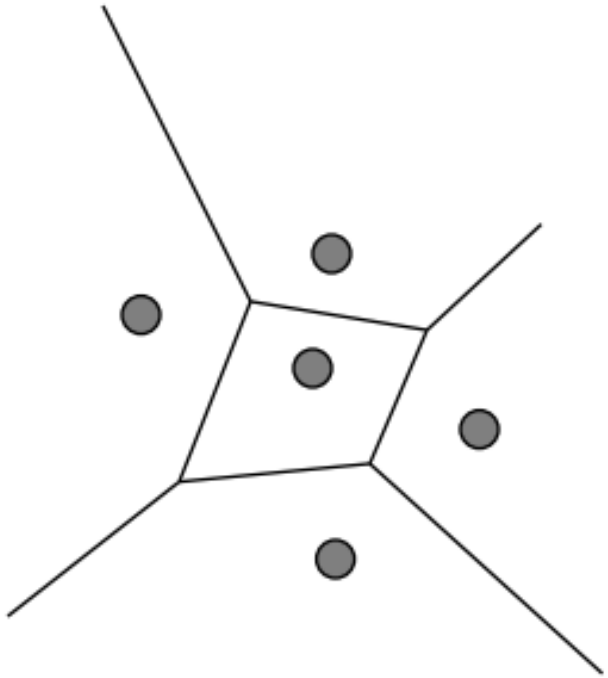


Figure: Voronoi Diagram

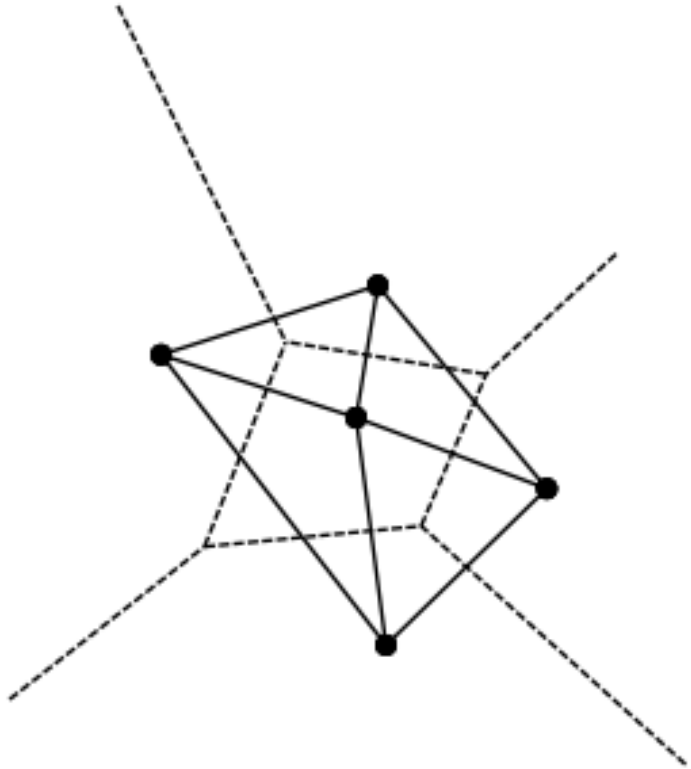


Figure: Delaunay Triangulation

Relationship with Voronoi diagram

For each Voronoi diagram there is a unique Delaunay triangulation and vice versa.

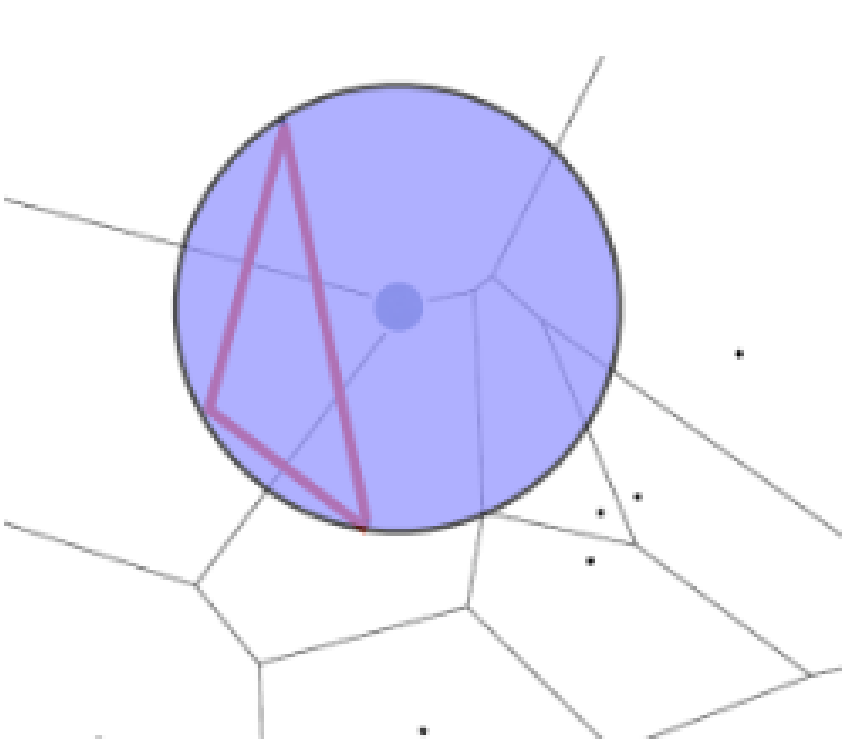
The nodes (or generators) of the Voronoi diagram are the vertices of the Delaunay triangulation.

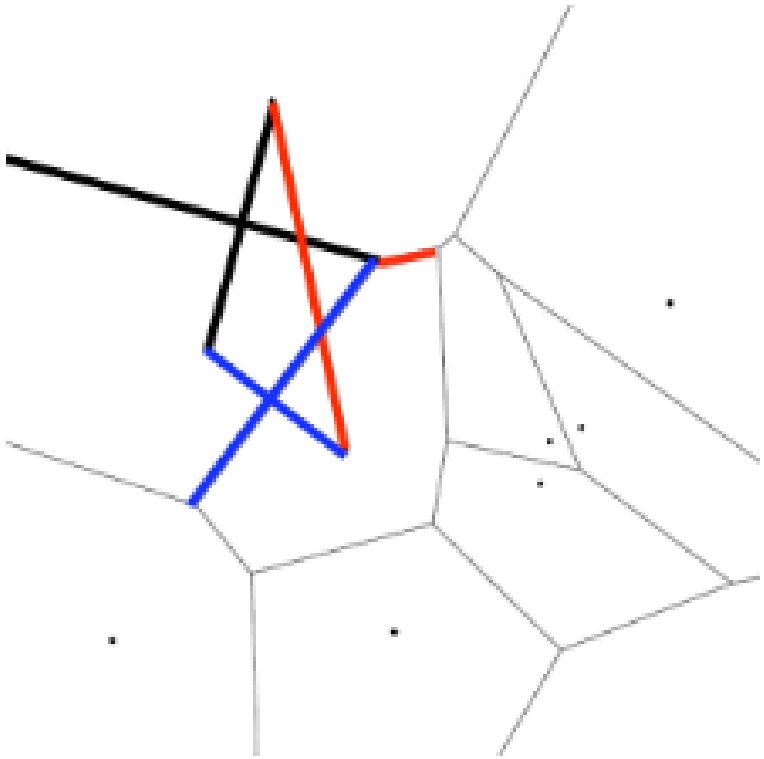
Each triangle of the Delaunay triangulation is associated with one vertex of the Voronoi diagram, which is located at the center of the circumscribed circle.

Each cell edge of the Voronoi diagram is uniquely associated with one cell edge of the Delaunay triangulation.

Each pair of edges are orthogonal, but do not necessarily intersect. If the pair of edges do intersect then the intersection point will bisect the line segment connecting generators.

Relationship with Voronoi diagram





Contents

Generating Voronoi Diagrams - Brute Force Algorithm

Easy to implement.

Poor performance in later stages as number of vertices increase.

Complexity: $O(n^3)$ - Calculation of every cell with N-1 sides (and N number of seed points), takes $O(n^2)$ time complexity.

Sometimes removes points that would contribute to the Voronoi cell.

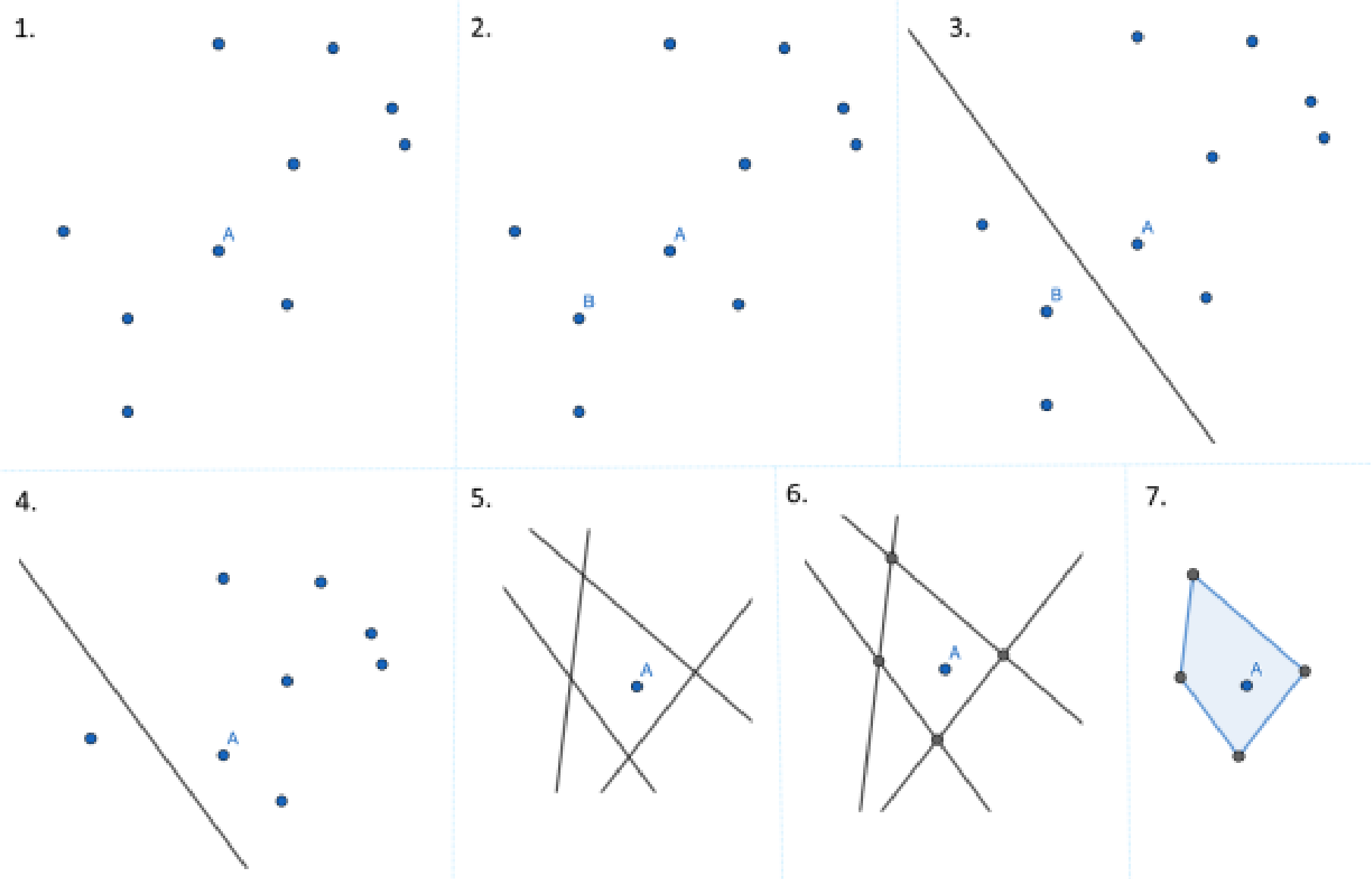
Can be easily extended to higher dimensions.

Generating Voronoi Diagrams - Brute Force Algorithm

Steps

1. Pick a seed point from the list of points. Let's call it point A.
2. Calculate the distance from A to every other point.
3. Pick the point closest to A. Let's call it point B.
4. Find the perpendicular bisector between points A and B.
5. Remove all the points on the other side of the bisector that are farther away from the bisector than B.
6. Remove B.
7. Repeat steps 3-6 until no more points remain.
8. Find the intersection points of the bisectors.
9. Connect the intersection points clockwise or counterclockwise to create a polygon.

Generating Voronoi Diagrams - Brute Force Algorithm



Generating Voronoi Diagrams - Fortune's Algorithm

Generates Voronoi diagram in 2D.

Time Complexity: O($nlogn$) - There are O(n) events to process and O(logn) time to process each event from binary search tree or priority queue.

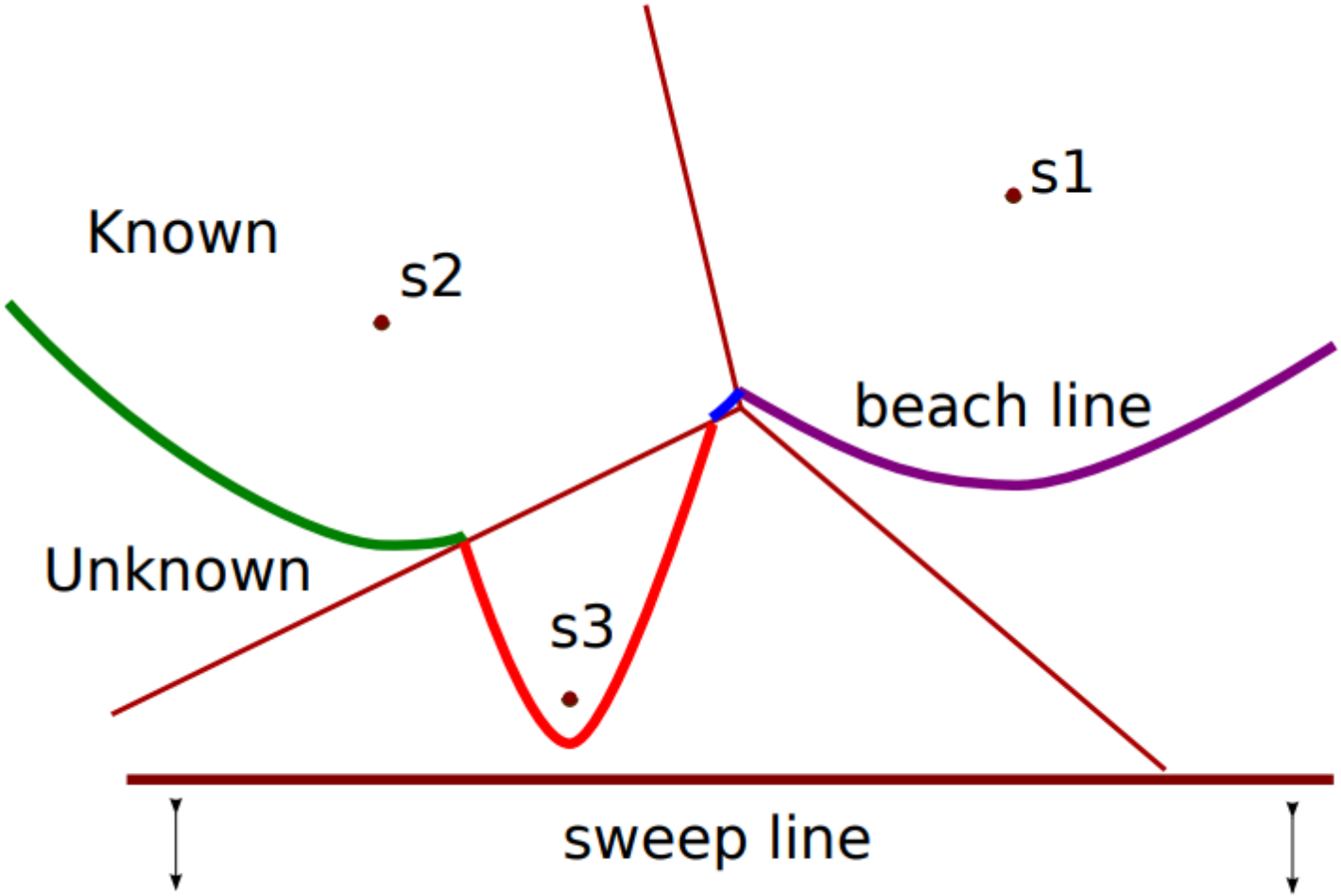
Space Complexity: O(n)

Maintaining Data Structures:

- A binary search tree to describe the combinatorial structure of beach line.
- A priority queue listing potential future events that could change the beach line structure.

Generating Voronoi Diagrams - Fortune's Algorithm

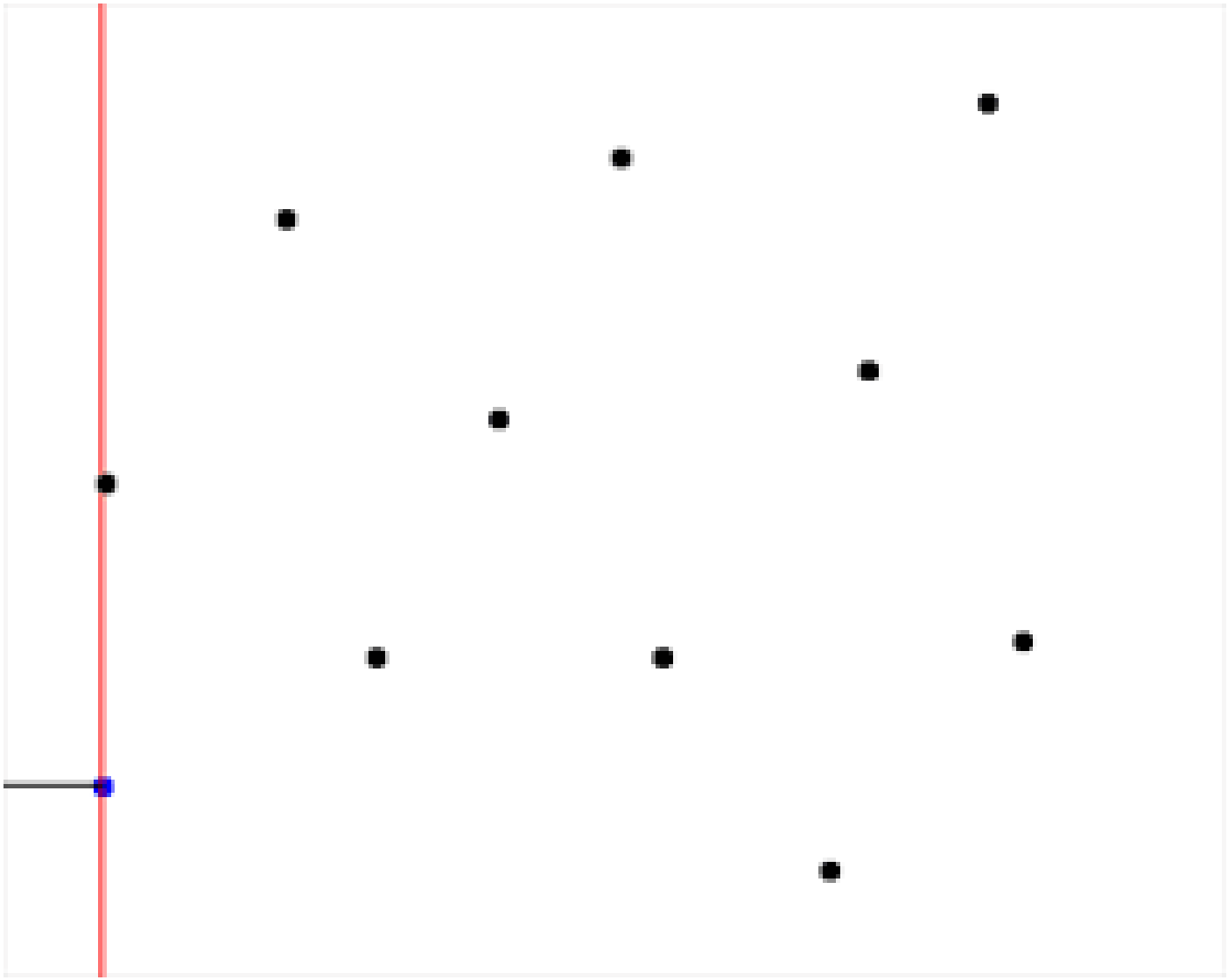
Fortune's algorithm is sweep line method



Steps

1. Fill the event queue with site events for each input site.
2. While the event queue still has items in it:
3. If the next event on the queue is a site event add the new site to the beachline.
4. Otherwise it must be an edge-intersection event remove the squeezed cell from the beachline.
5. Cleanup any remaining intermediate state.

Generating Voronoi Diagrams - Fortune's Algorithm



Generating Voronoi Diagrams - Fortune's Algorithm

Disadvantage of Fortune's Algorithm : Lack of ability to add new vertices later on.

Need of recalculating the entire diagram every time.

Bad choice for real-time destruction calculation.

Generating Voronoi Diagrams - Bowyer-Watson Algorithm

Sacrificing calculation time, **Bowyer-Watson algorithm** has the ability to dynamically change whenever a new point is added.

It is used to calculate the Delaunay triangulation. Voronoi diagram can be easily derived from that.

Worst Case Complexity: $O(n^2)$

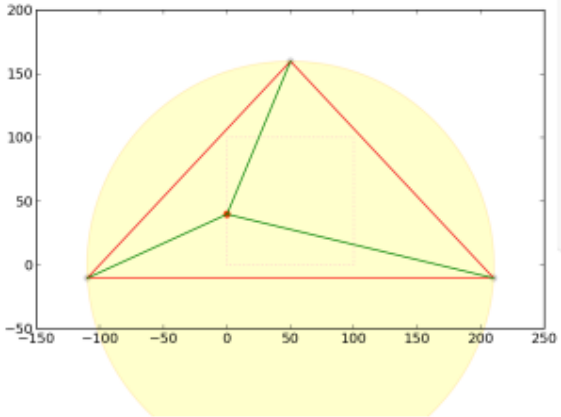
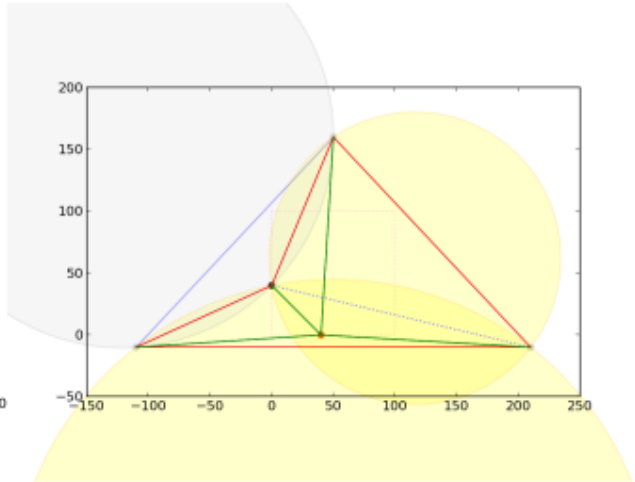
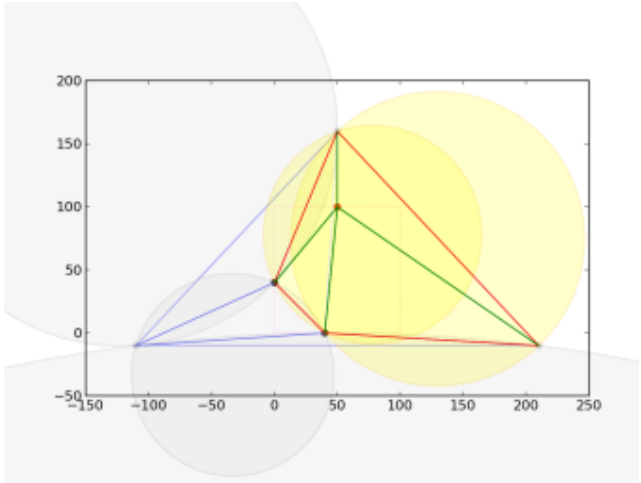
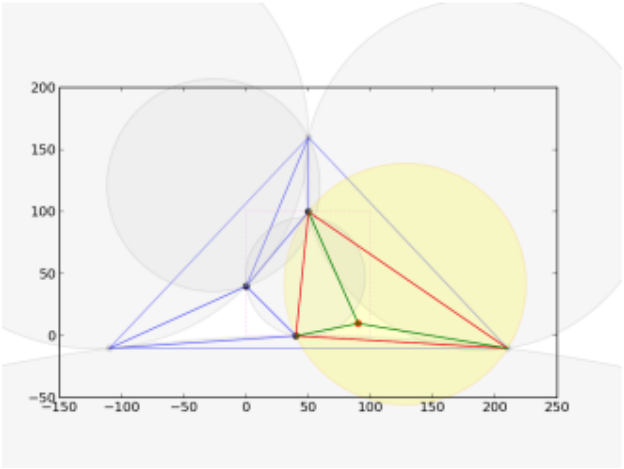
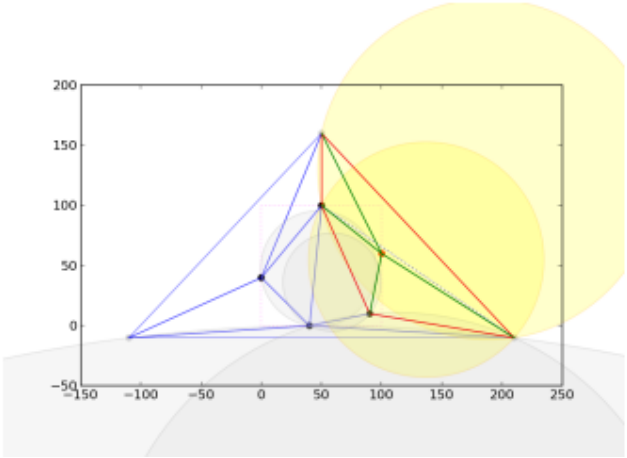
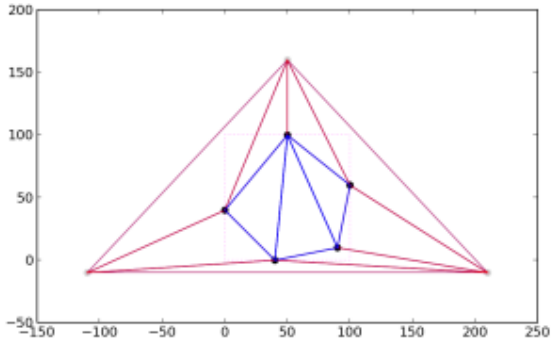
but with multiple optimization techniques, O(nlogn) can be reached.

Generating Voronoi Diagrams - Bowyer-Watson Algorithm

Steps

1. Create a triangle that contains all the points to be processed.
2. Add a point to the triangulation
3. Find all the triangles whose circumcircle contains the new point.
4. Remove these triangles.
5. Connect the vertices of the remaining hole to the new point.
6. Repeat 2-5 until all points are processed.
7. Remove all triangles that are connected to the original triangle.

Generating Voronoi Diagrams - Bowyer-Watson Algorithm

Contents

Clustering

- What is it ?
- What is K-Means algorithm ?
- How is associated with Vornoi Diagrams ?
- Can we apply Voroni Digrams on K-Meams as an optimization ?

Clustering - What is it?

- **Clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).
- **Unsupervised Learning Technique** : This means that we don't know from the start which are the categories to group the data by.
- We can apply Clustering in a wide variety of fields such as Data Mining, Machine Learning, Pattern Recognition, Image Analysis etc.

K-Means Algorithm

Result: $Clusters = \{c_1, ..., c_k\}, k = TotalClusters$

Step 1: Select k initial cluster centers $\{c_1, ..., c_k\}$ randomly from the given n points $\{x_1, ..., x_k\}$, $k \leq n$;

Step 2: Assign each point x_i, $i = \{1, ..., n\}$ to the cluster C_j corresponding to the cluster center c_j, for $j = \{1, ..., k\}$ iff $||x_i - c_j|| \leq ||x_i - c_p||$, $i = \{1, ..., k\}$ and $j \neq p$;

Step 3: Compute new cluster centers $c_1^*, ..., c_k^*$ with :

$$c_i^* = 1/n_i * \sum_{x_j \in C_i} x_j \ for\ i = \{1, ..., k\}$$

where n_i is the number of data points belonging to the cluster C_i;

Step 4: if $c_i^* = c_i$, $\forall i = 1, ..., k$, then terminate else goto Step 2;

Algorithm 1: K-Means

Theorem

If we let n, n_e, and n_v, be the number of generator points, Voronoi edges and Voronoi vertices of a Voronoi diagram, respectively. We find that :

$$(1)\ n_v \text{ - } n_e + \text{n} = 1$$

If P has more than 3 elements, then $n_e \leq 3n - 6$ *and* $n_v \leq 2n - 5$

Proposed Initialization Method for K-means

Given the set of n data points S and the required number of clusters k, the Voronoi diagram Vor(S) is constructed first.

We next need to sort all the vi verices on a non increasing order of their radius of largesr empty Voronoi circles an store them into sorted vertex[i].

We maintain 2 arrays:

- Test [ ] (to store all the points on the circumference of Voronoi circle)
- ccenter [ ] (to hold the partial or final initial cluster centers) which are initially empty. We start the iteration with the largest radius Voronoi circle.

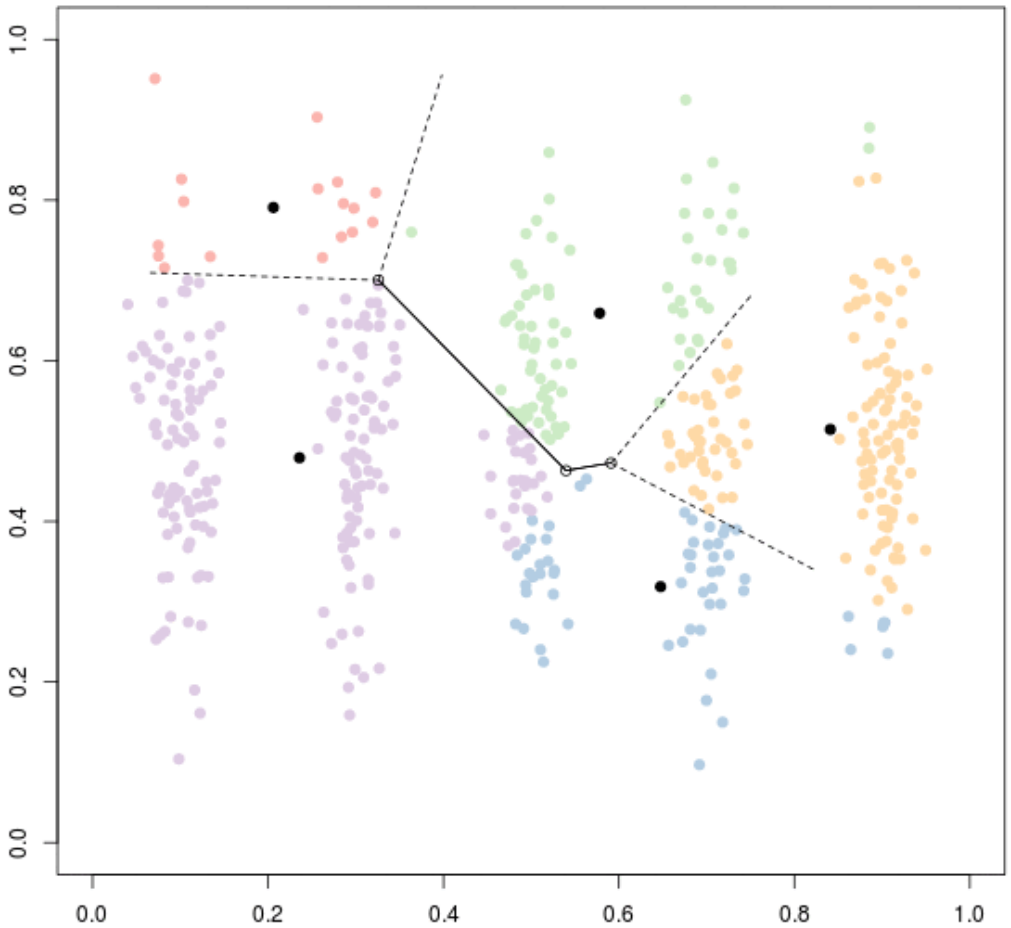
Proposed Initialization Method for K-means

In every iteration ($2n * 5$ maximum), we:

1. Store all the points on the boundary of CirS(sorted vertex [i]) in Test[]. If the distance between any 2 points of Test [ ] is less than the radius of CirS(sorted vertex [i]), remove any one of them from Test[].
2. Check the distance between every point of Test [ ] and all the points of ccenter [ ]. If the distance between any two points of Test [ ] and ccenter [ ] is less than the radius of CirS(sorted vertex [i]) then also remove the point from Test [ ].
3. Store all the remaining points of Test [ ] into ccenter [ ].

   At the end of the final iteration, the set of initial cluster centers are stored in ccenter [ ] which is then provided to the K-means clustering algorithm.

K-means Method



Contents

R - Trees

R-trees are tree data structures used for spatial access methods.

- Key idea: Group nearby objects and represent them with their **minimum bounding rectangle** in the next higher level of the tree.
- Balanced search tree.
- Organizes data in pages.
- Designed for storage on disk.
- Each page can contain a maximum number of entries. It also
- Guarantees a minimum fill.

R - Trees

Can be used to:

- Quickly arrive to the neighborhood of the result set.

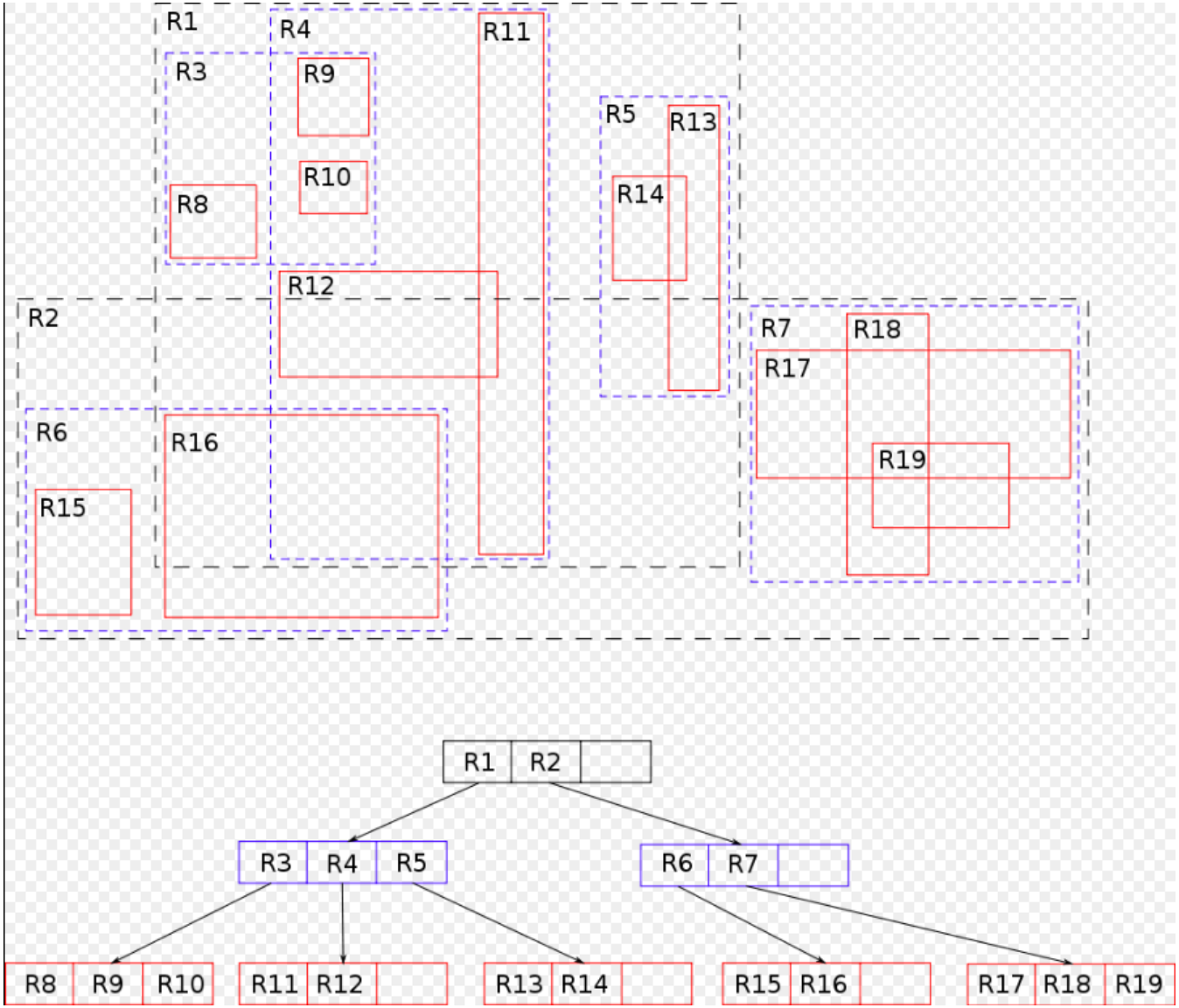
But

- the R-tree nodes intersecting with this neighborhood need to be investigated to find all the members of the result set. - only a small subset of them belongs to the actual result set. -

Difficulty:

- Building an efficient **balanced** tree that the rectangles do not cover too much empty space and do not overlap too much.

R - Trees


R1
R4
R11
R3
R9
R5
R13
R10
R14
R8
R12
R2
R7
R18
R17
R6
R16
R19
R15
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19

Need of Voronoi Diagram

Voronoi Diagrams:

- Are extremely efficient in exploring NN search region.

But :

- the arrival to this region is very slow.

Vor - Tree

What is it ?

- An Tree that that incorporates Voronoi diagrams into R-tree, benefiting from both.

Definition of Vor - Tree

Suppose that we have stored all the data points of set P in an R-tree. We have pre-built the Voronoi diagram of P.

Each leaf node of R-tree stores a subset of data points of P. The leaves also include the data records containing extra information about the corresponding points.
In the record of the point p, we store the pointer to the location of each Voronoi neighbor of p (V N(p)) and also the vertices of the Voronoi cell of p (V (p)).

The above instance of R-tree built using points in P is the VoR-Tree of P.

R - Trees

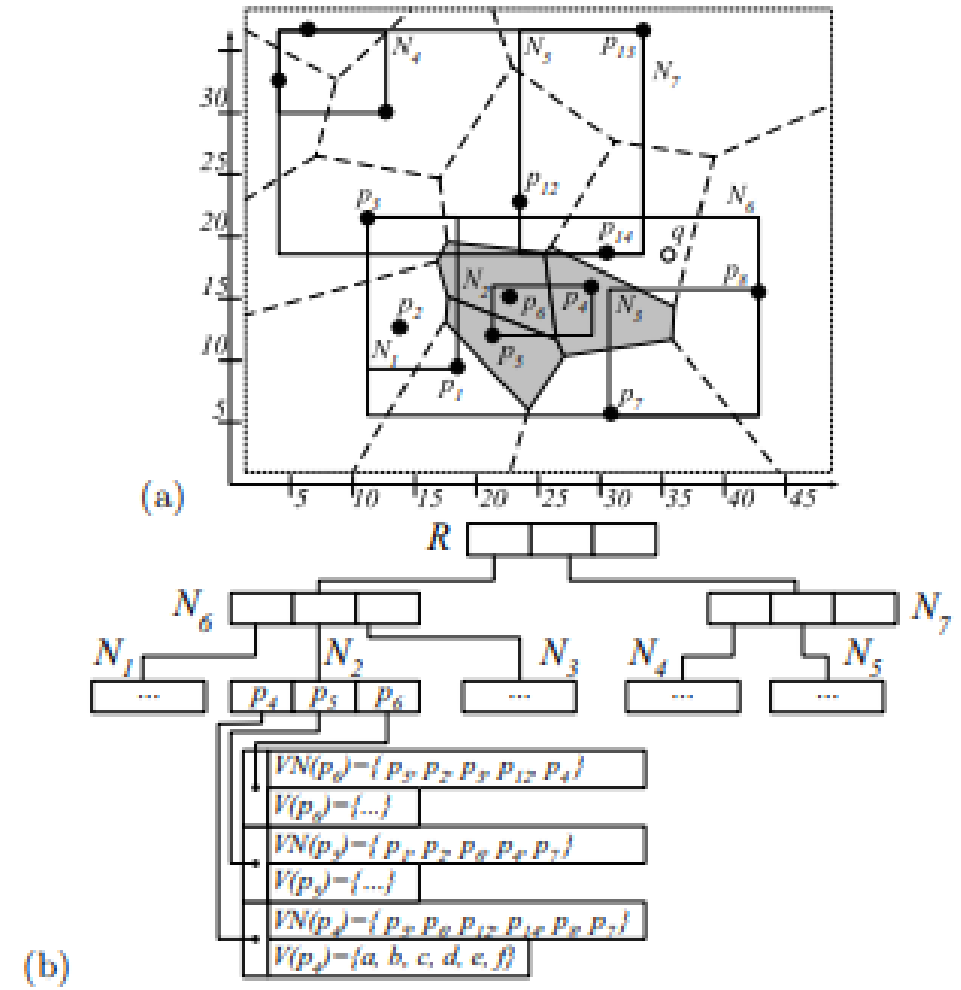


Figure 2: a) Voronoi diagram and b) the VoR-Tree of the points shown in Figure 11

Contents


Franz Aurenhammer, Rolf Klein, *Voronoi Diagrams*, 2018.


Fortune Steven, *A sweepline algorithm for Voronoï diagrams*, 1987.


Voronoi Tessellations: http://datagenetics.com/blog/may12017/index.html


D.T.Lee and B.J.Schachter, *Two Algorithms for Constructing a Delaunay Triangulation*, 2012.


Damodar Reddy, Prasanta K. Jana, *Initialization for K-means clustering using Voronoi diagram*, 2012.


Antomn Guttman, *R-TREES. A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING*, 1984


Mehdi Sharifzadeh, Cyrus Shahabi, *VoR-tree: R-trees with Voronoi diagrams for efficient processing of spatial nearest neighbor queries*, 2010.