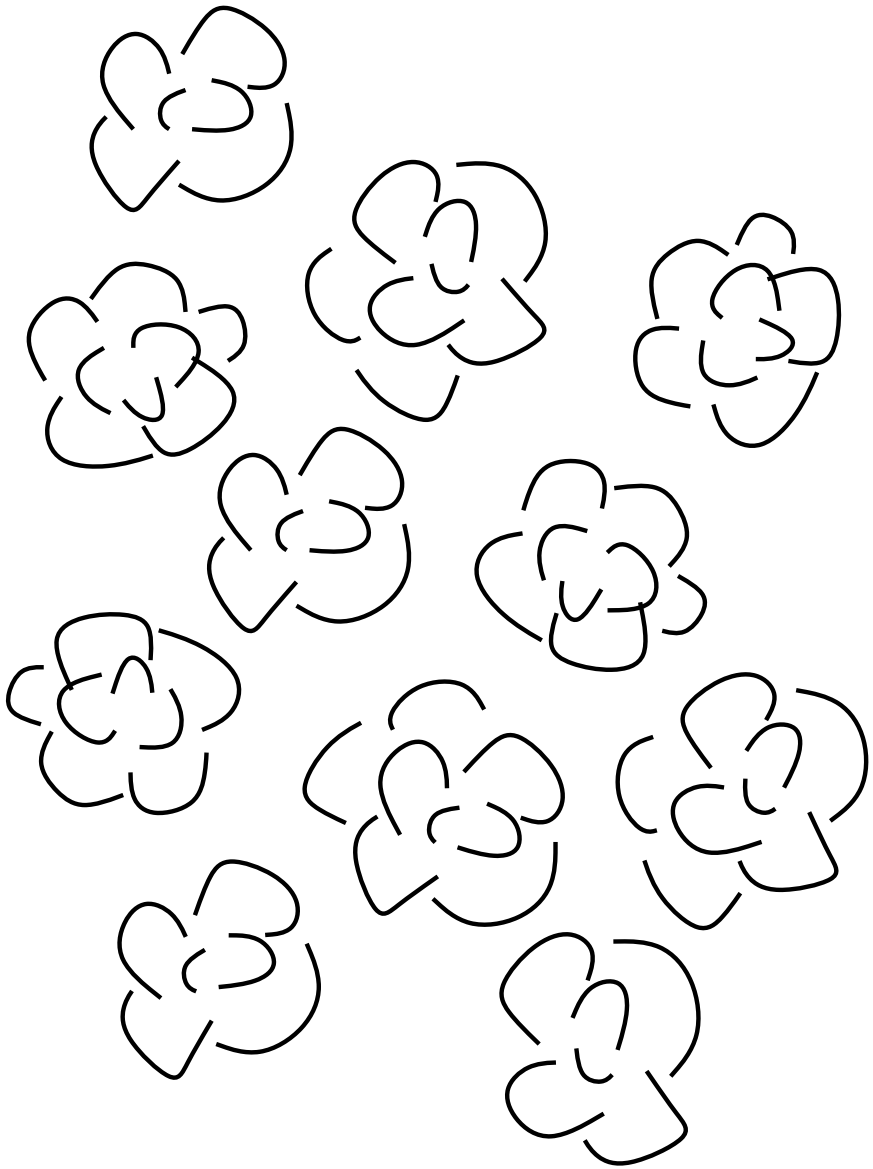
Voronoi diagrams

Computational Geometry

Lecture 10: Voronoi diagrams

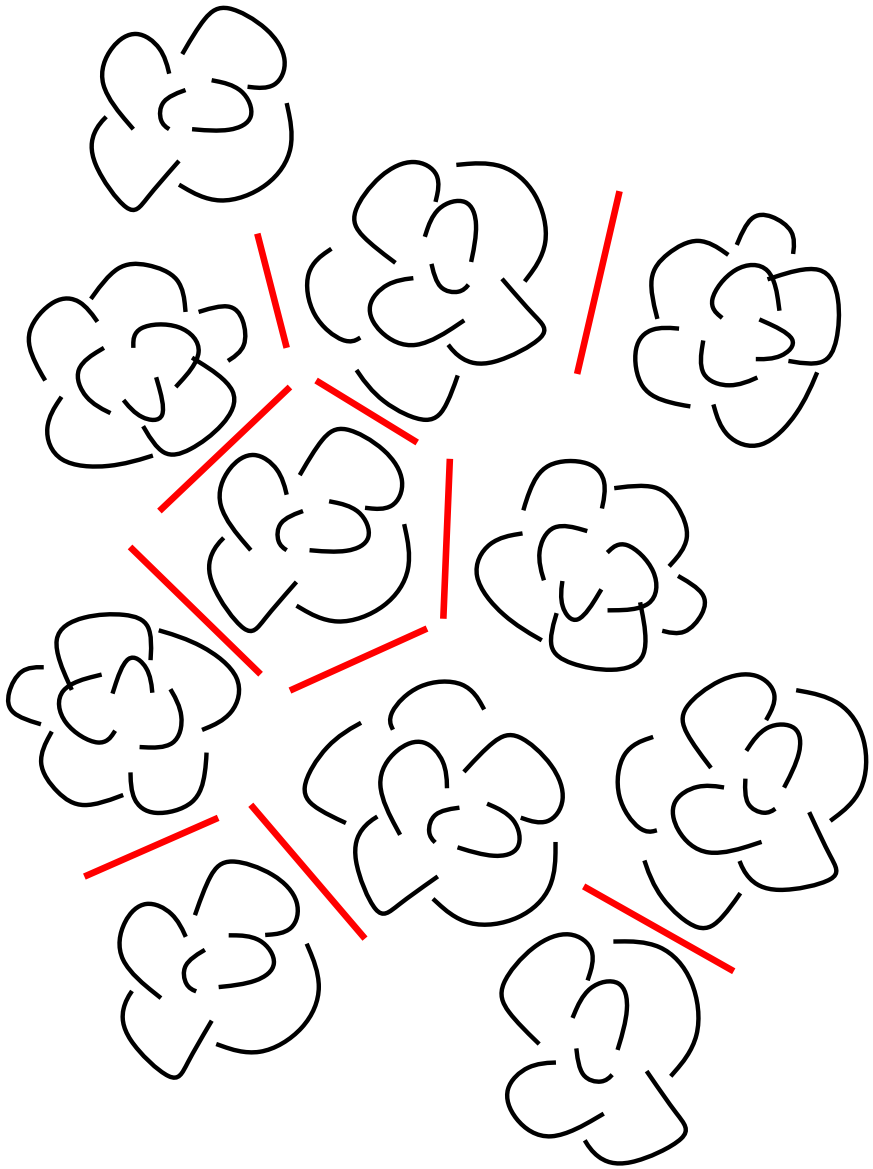
Voronoi diagram

Given some trees, seen from above, which region will they occupy?



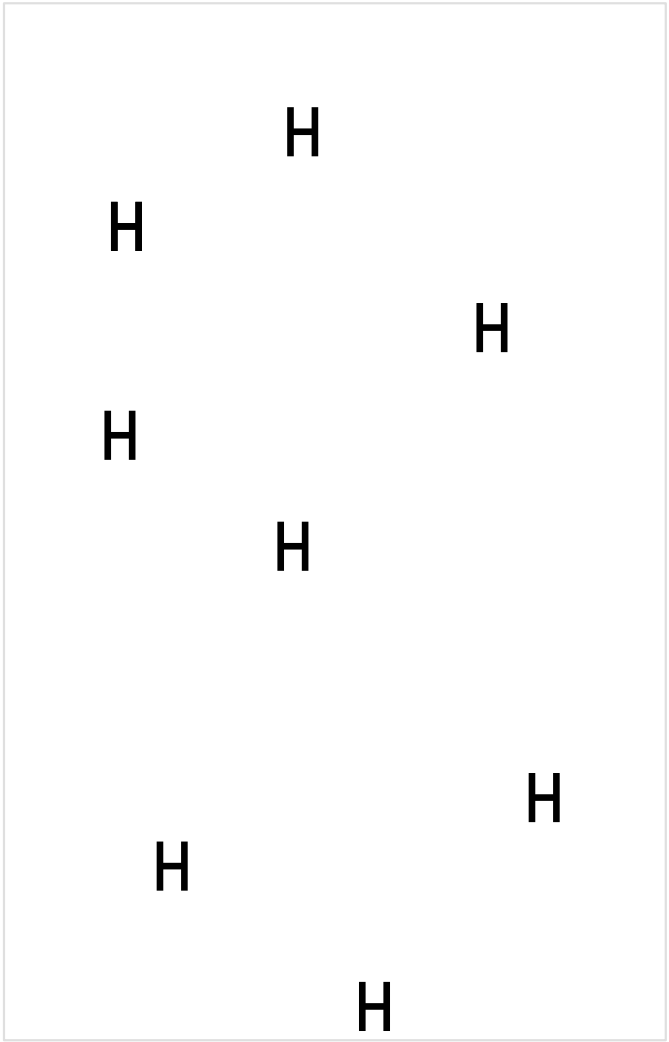
Voronoi diagram

Given some trees, seen from above, which region will they occupy?



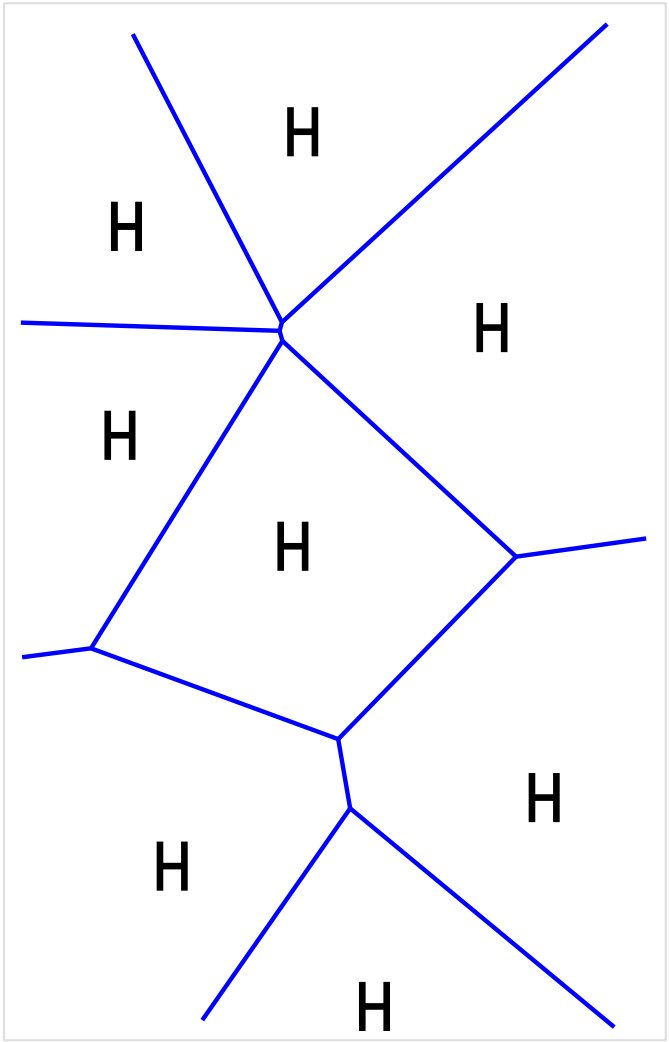
Voronoi diagram

Given ambulance posts in a country, in case of an emergency somewhere, where should the ambulance come from?



Voronoi diagram

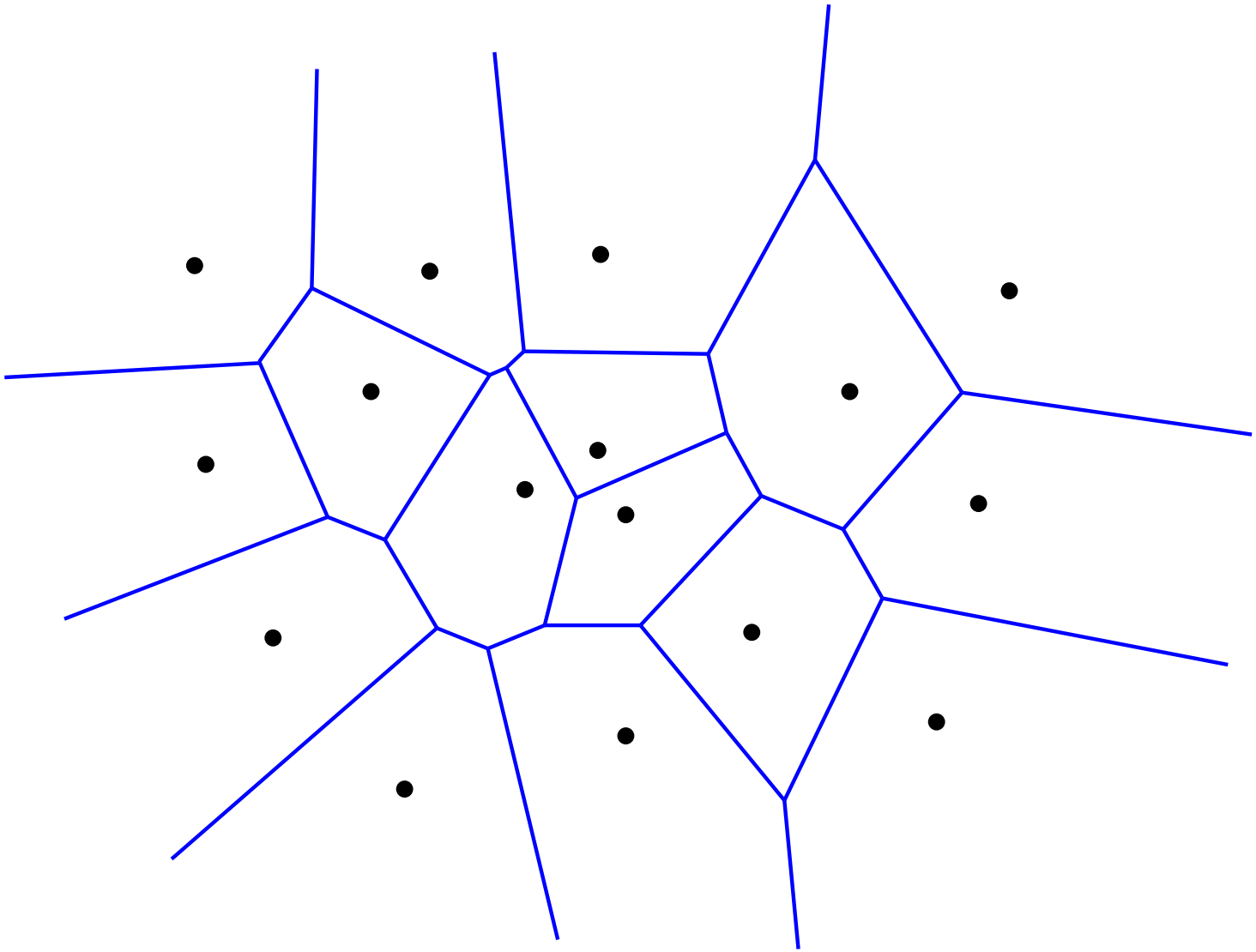
Given ambulance posts in a country, in case of an emergency somewhere, where should the ambulance come from?



Voronoi diagram

Voronoi diagram induced by a set of points (called sites): Subdivision of the plane where the faces correspond to the regions where one site is closest

Voronoi diagram

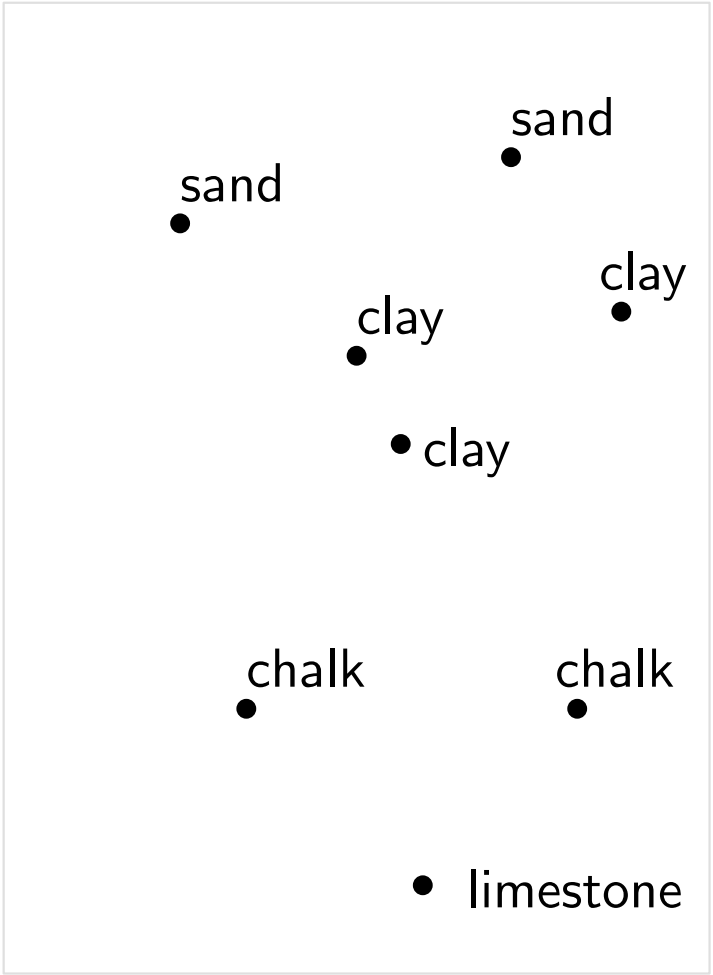


Voronoi diagram

Question: Why is the Voronoi diagram not really a subdivision?

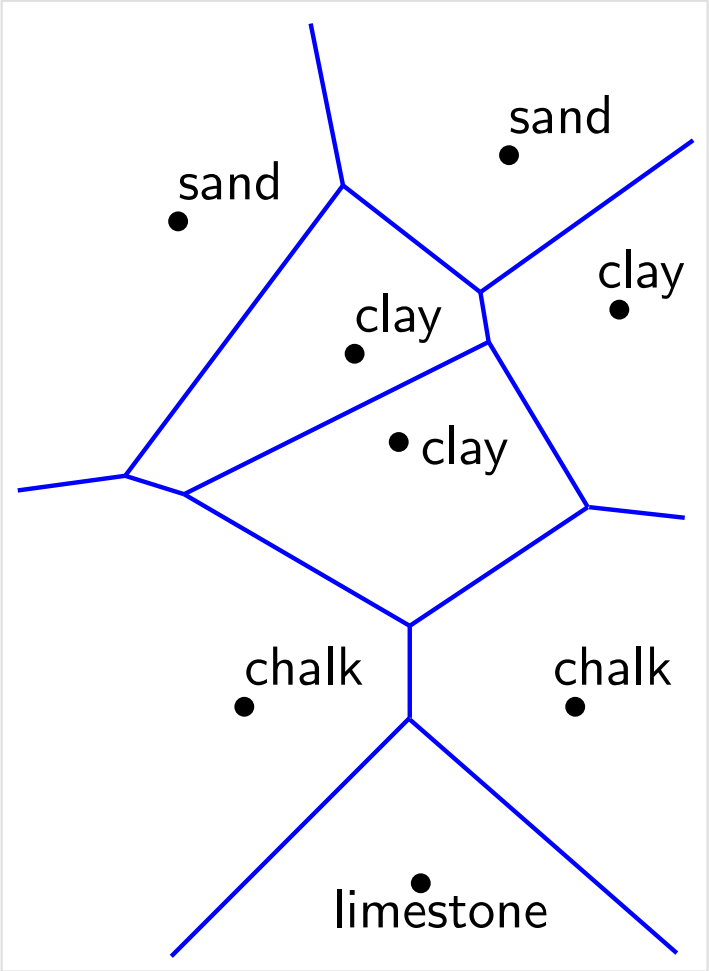
Spatial interpolation

Suppose we tested the soil at a number of sample points and classified the results



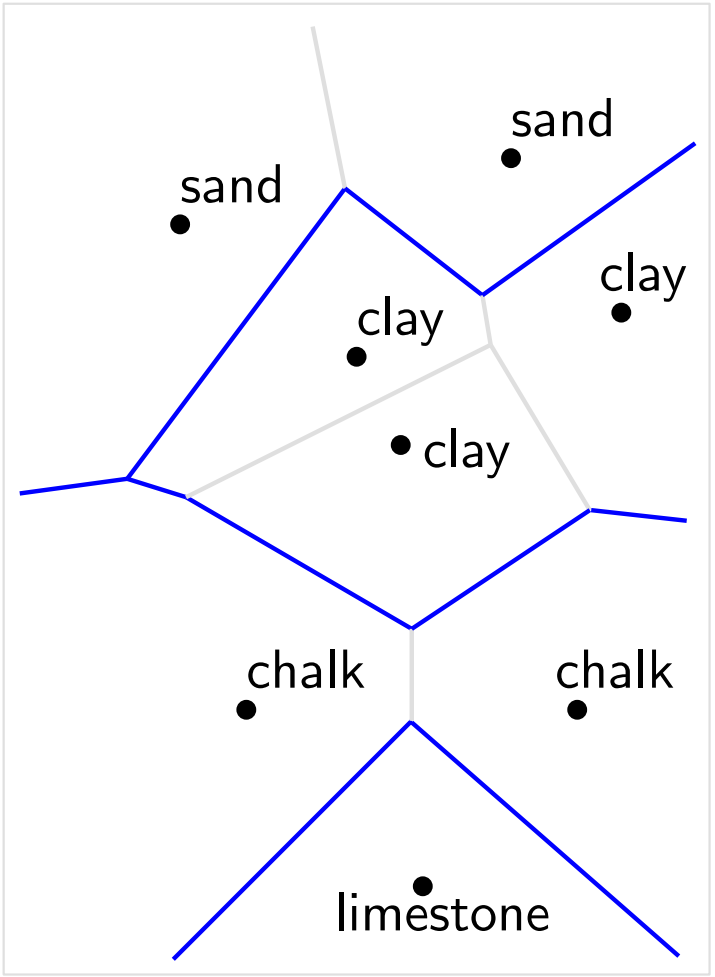
Spatial interpolation

Suppose we tested the soil at a number of sample points and classified the results



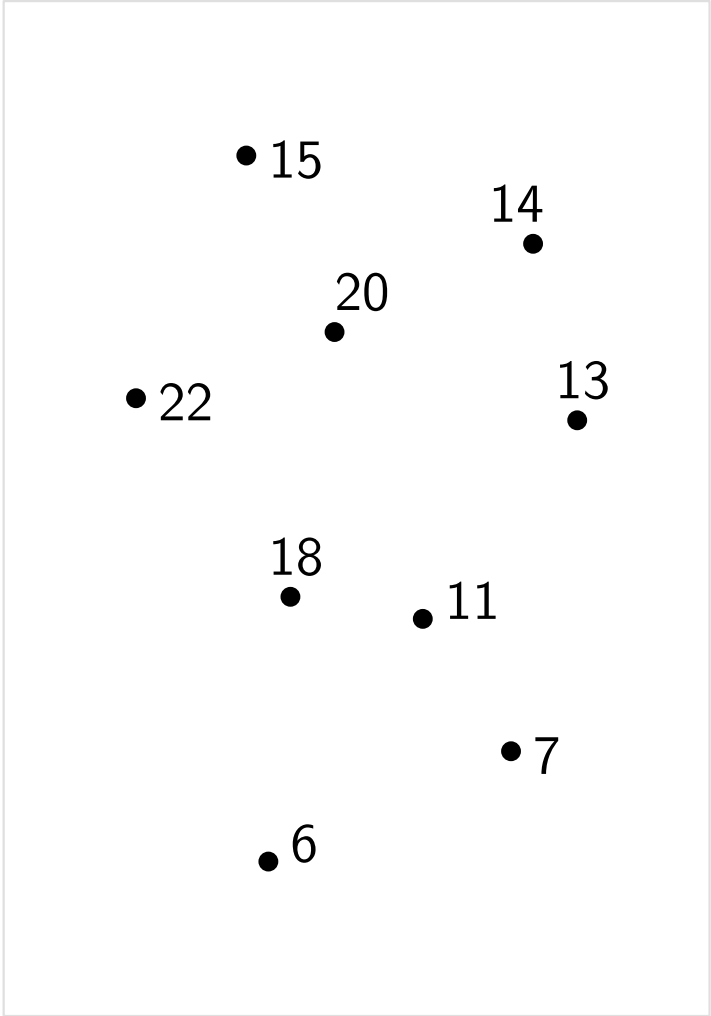
Spatial interpolation

Suppose we tested the soil at a number of sample points and classified the results



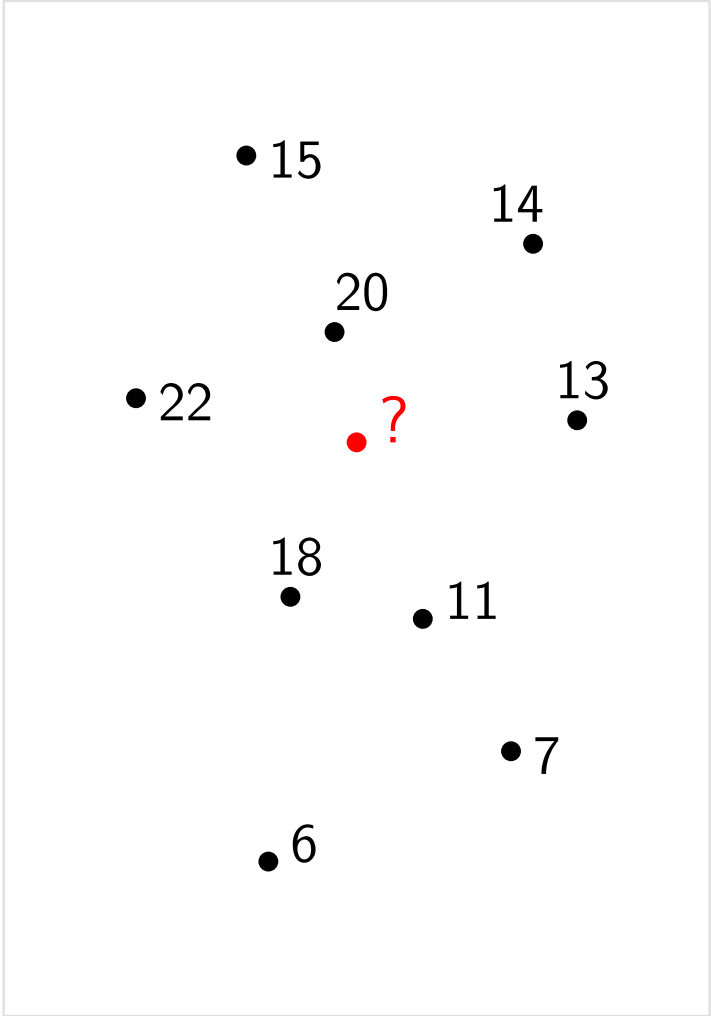
Spatial interpolation

Suppose we measured the lead concentration at a number of sample points



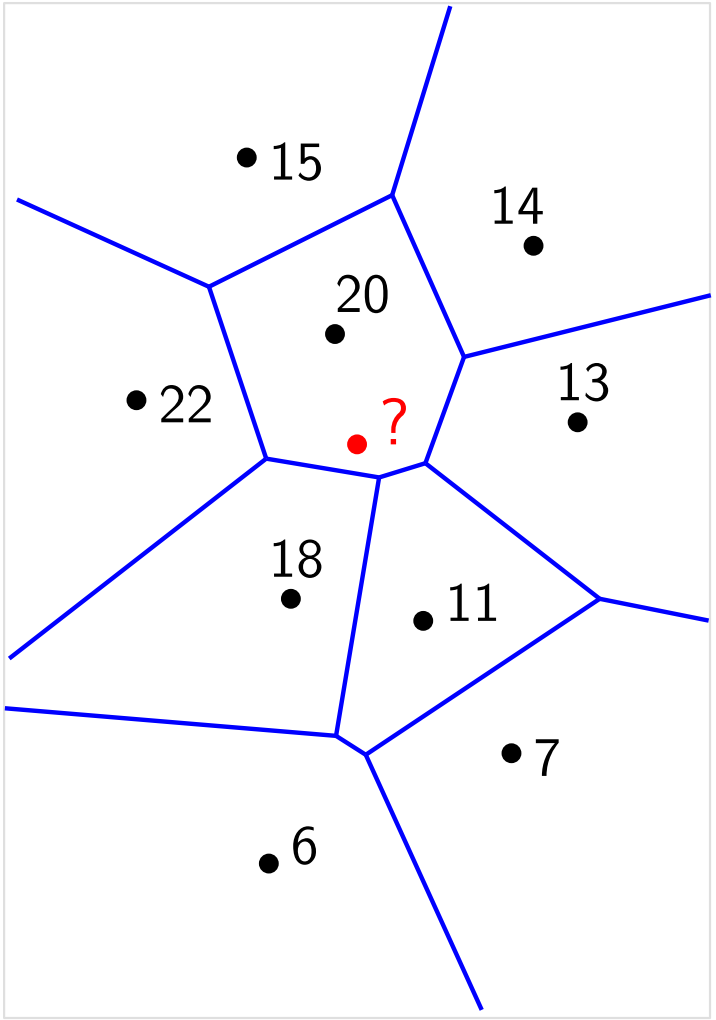
Spatial interpolation

Suppose we measured the lead concentration at a number of sample points



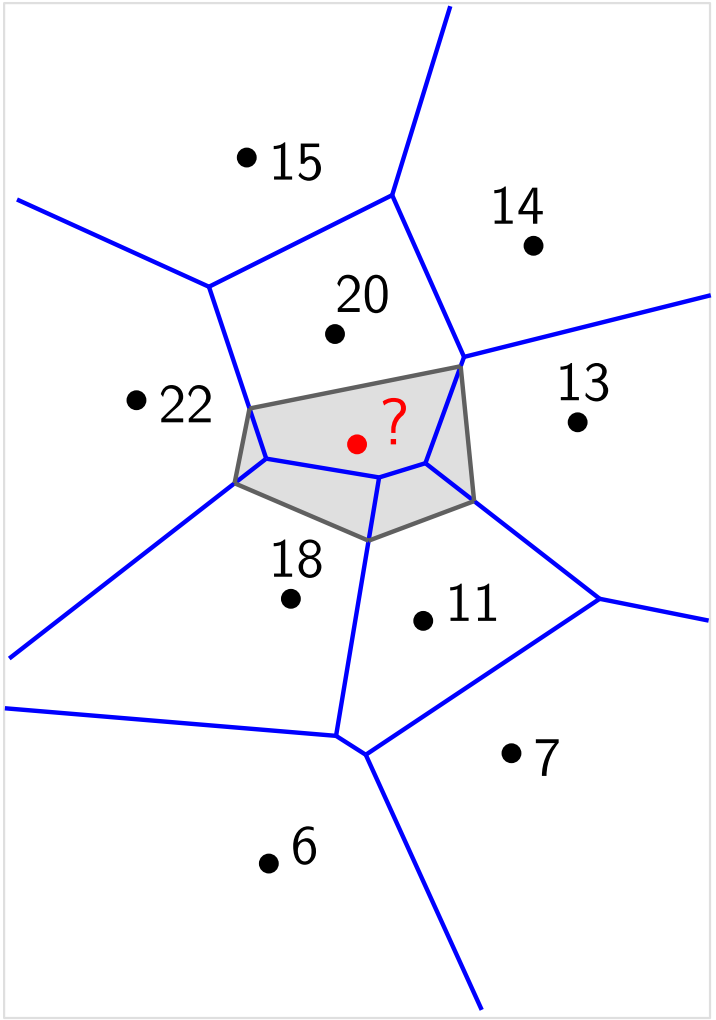
Spatial interpolation

Suppose we measured the lead concentration at a number of sample points



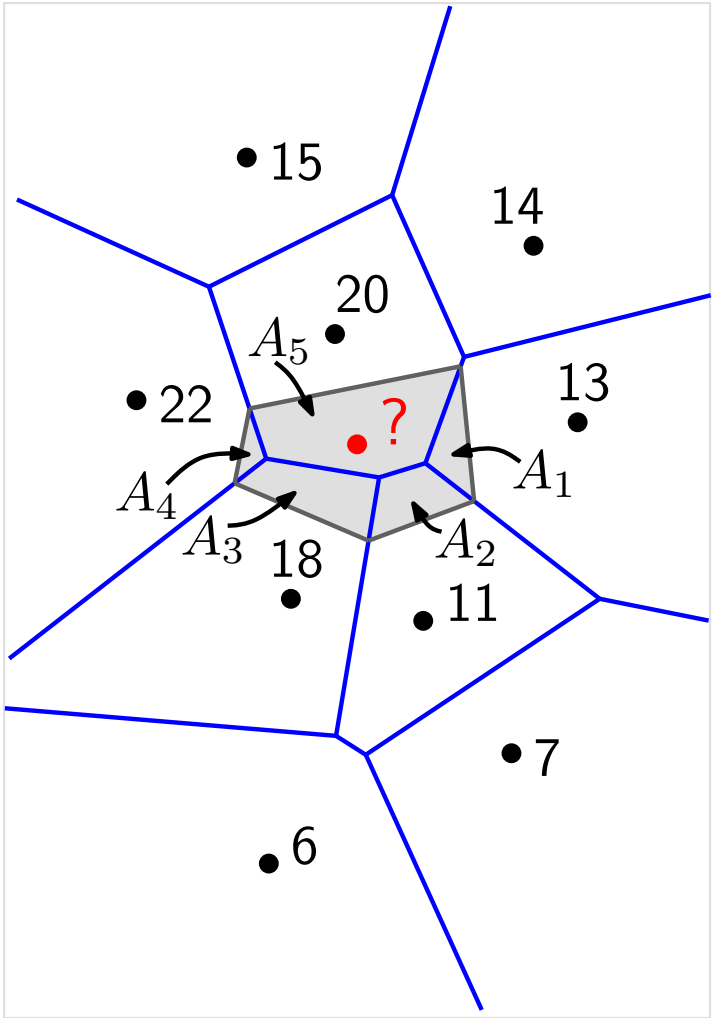
Spatial interpolation

Suppose we measured the lead concentration at a number of sample points



Spatial interpolation

Suppose we measured the lead concentration at a number of sample points

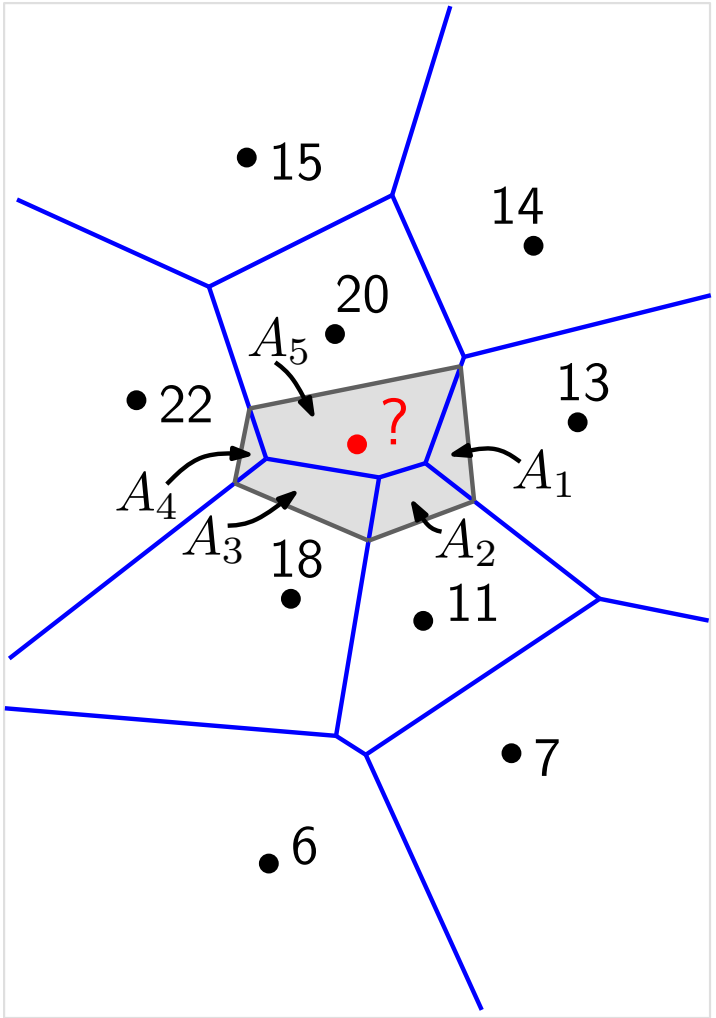


Spatial interpolation

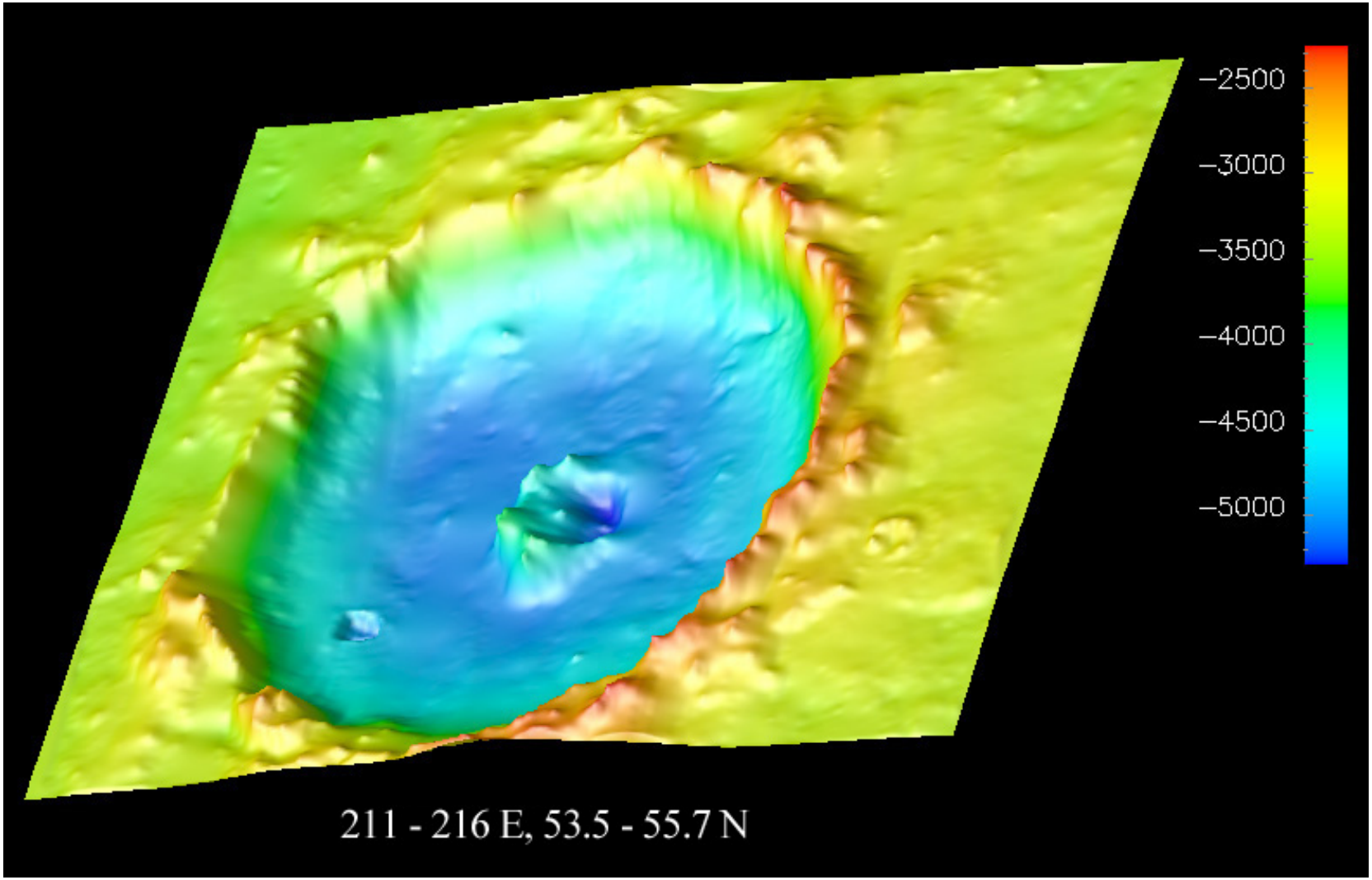
Let $A_T = A_1 + A_2 + \cdots + A_5$

The interpolated value is

$$\frac{A_1}{A_T} 13 + \frac{A_2}{A_T} 11 + \cdots + \frac{A_5}{A_T} 20$$



Spatial interpolation



Crater on Mars generated by natural neighbor interpolation

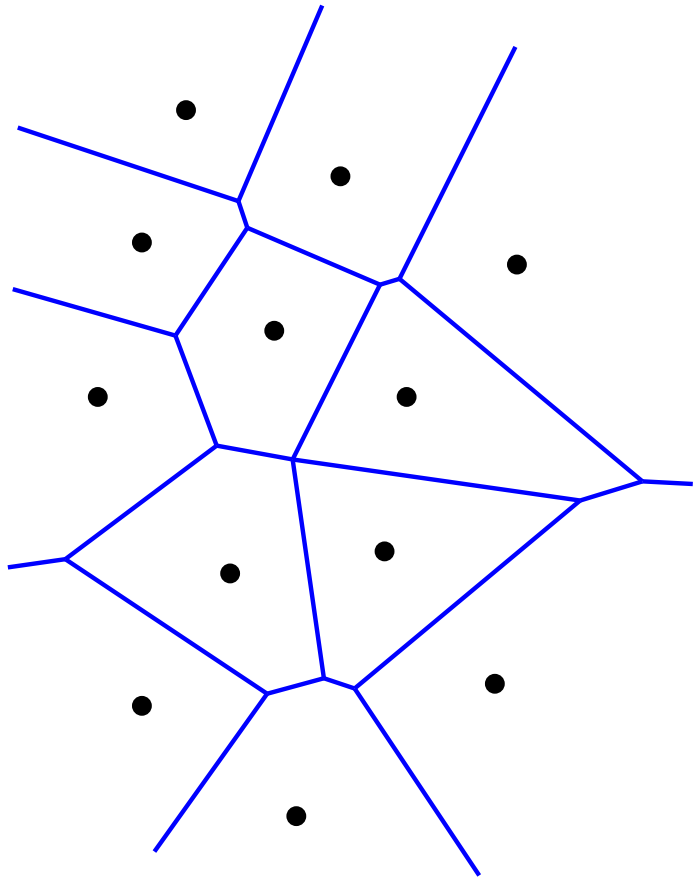
Some observations

Edges are parts of bisectors

Some edges are half-infinite

Some cells are unbounded

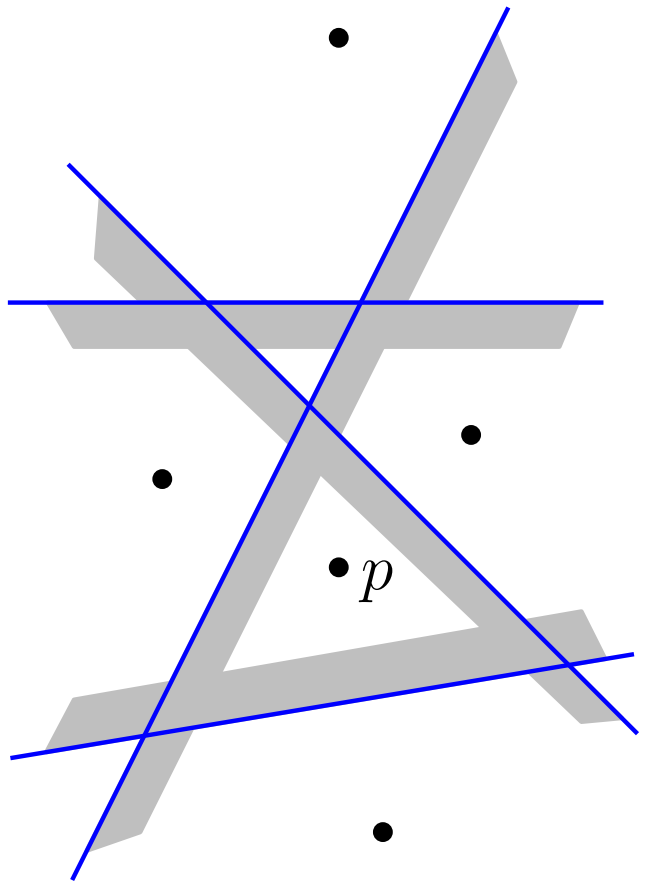
Question: Which ones?



Some observations

Every Voronoi cell is the intersection of $n-1$ half-planes, if there are n sites

$\Rightarrow$ all cells are convex and have up to $n-1$ edges in the boundary



Structure

The Voronoi diagram of n sites has the following structure:

- If all n sites lie on a line, then the Voronoi cell boundaries are parallel lines, so the "graph" is disconnected
- Otherwise, the Voronoi cell boundaries form a connected "graph"

Complexity

Theorem: The Voronoi diagram on f sites in the plane has at most $2n-5$ Voronoi vertices and at most $3n-6$ Voronoi edges (including lines and half-lines)

Proof: If the sites are colinear, then it is trivial

Otherwise, we will use Euler's formula for planar graphs

Complexity

Euler's formula for planar graphs: A connected planar graph with n_v vertices, n_e edges, and n_f faces satisfies:

$$n_v - n_e + n_f = 2$$

However, a Voronoi diagram is not a proper graph

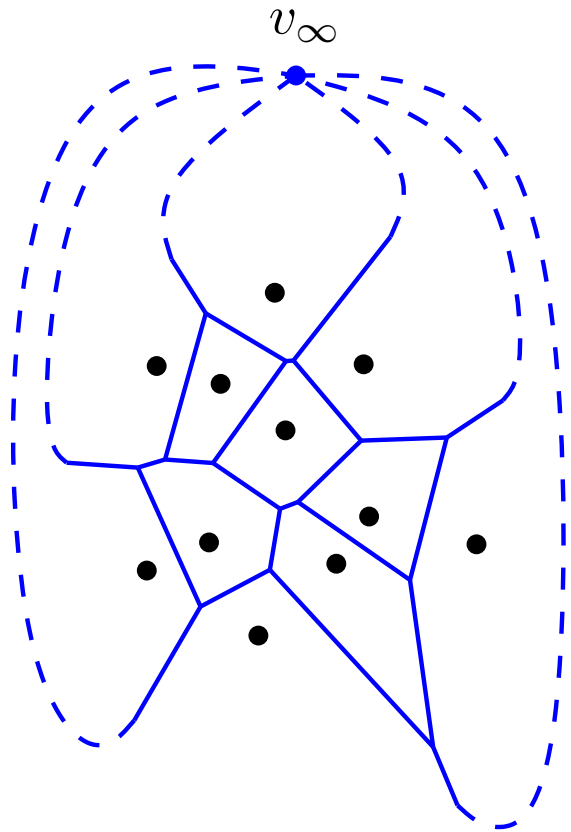
Complexity

We make it proper by connecting all half-infinite edges to a new vertex v_∞

$n_v =$ no. of Voronoi vertices $VV + 1$

$n_e =$ no. of Voronoi edges VE

$n_f =$ no. of Voronoi cells $= n$, the number of sites



Complexity

Substitution in Euler's formula $n_v - n_e + n_f = 2$ gives

$$(VV + 1) - VE + n = 2$$

Every edge is incident to exactly 2 vertices, and every vertex is incident to at least 3 edges

Sum-of-degree-of-all-vertices $= 2 \cdot VE$

Sum-of-degree-of-all-vertices $\geq 3 \cdot VV$

$$= 2 \cdot VE \geq 3 \cdot VV$$

Complexity

The combination of

$$(VV + 1) - VE + n = 2$$

and

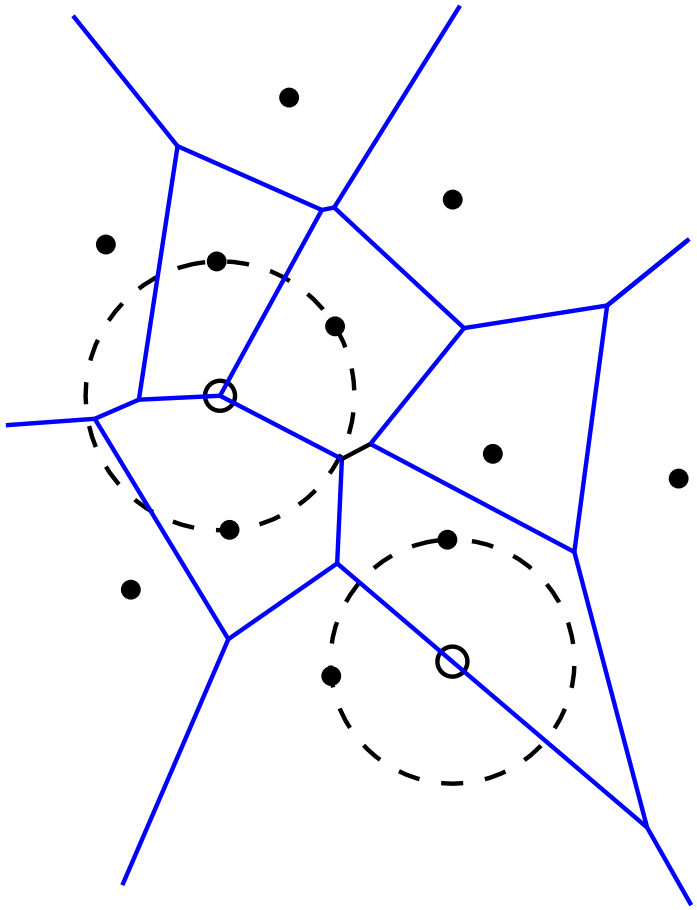
$$= 2 \cdot VE \geq 3 \cdot VV$$

gives the desired bounds $VV \leq 2n - 5$ and $VE \leq 3n - 6$

Empty circle property

Every Voronoi vertex is the center of an empty circle through 3 sites

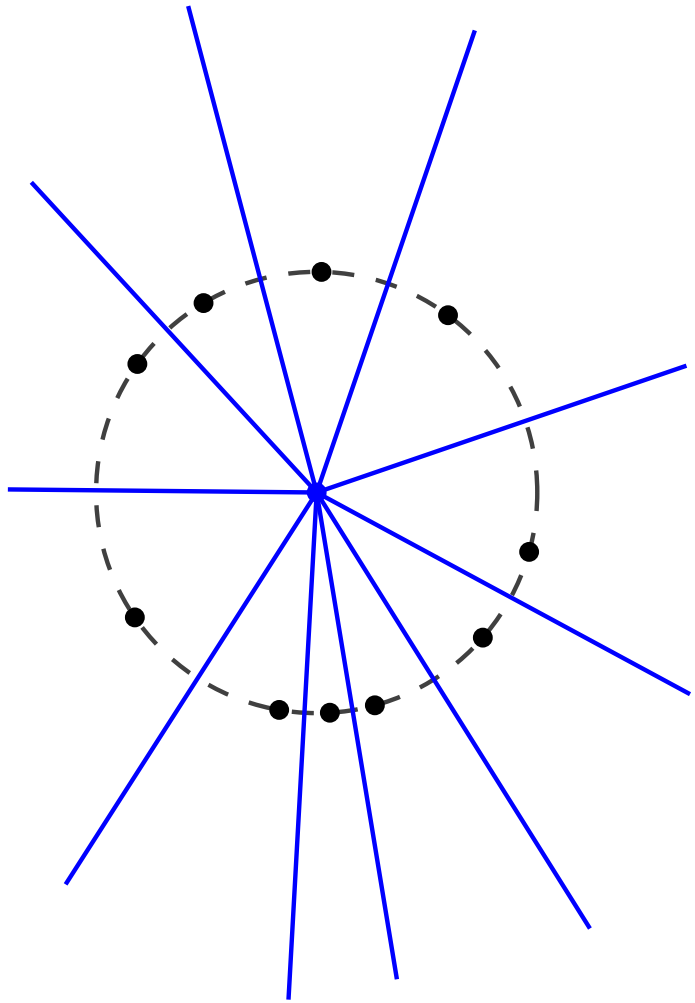
Every point on a Voronoi edge is the center of an empty circle through 2 sites



Degeneracies

All sites lie on a line

More than 3 points lie on a circle



Algorithms for Voronoi diagrams

Compute the intersection of $n-1$ half-planes for each site, and "merge" the cells into the diagram

Divide-and-conquer (1975, Shamos & Hoey)

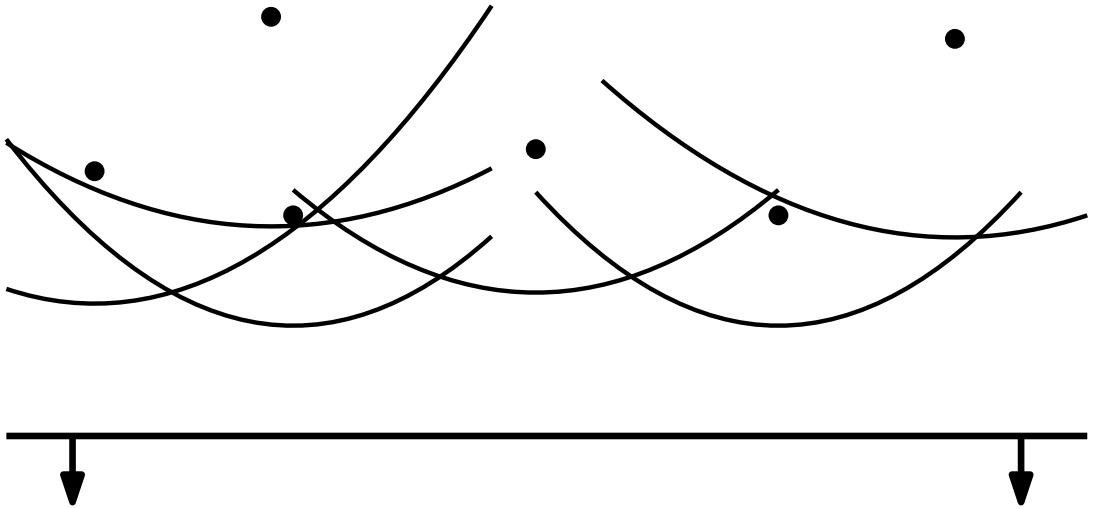
Plane sweep (1987, Fortune)

Randomized incremental construction (1992, Guibas, Knuth & Sharir)

Plane sweep for Voronoi diagrams

Plane sweep: Note that the Voronoi diagram above the sweep line may be affected by sites below the sweep line

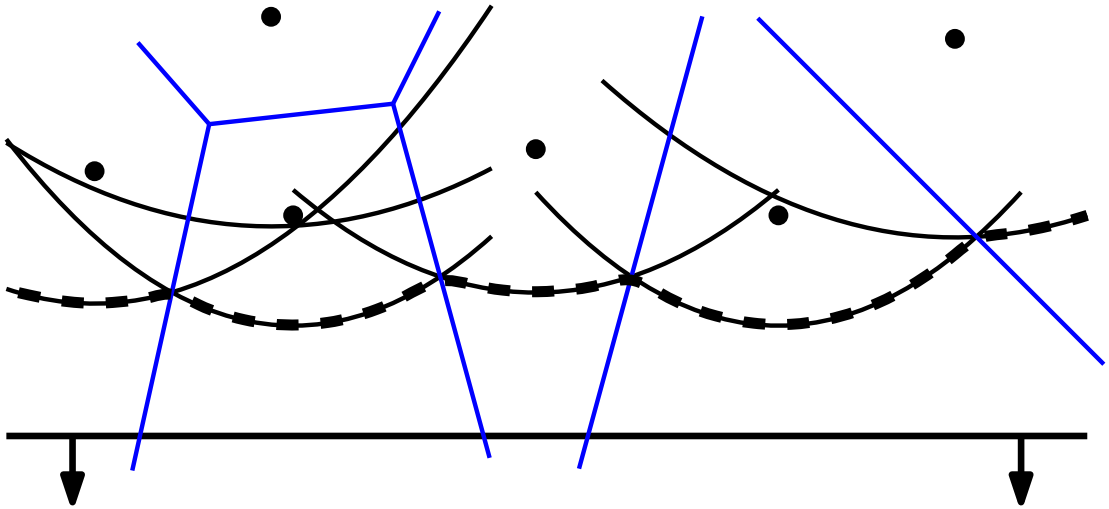
Maintain and grow the portion of Voronoi diagram above the sweep line that is *known for sure*



Plane sweep for Voronoi diagrams

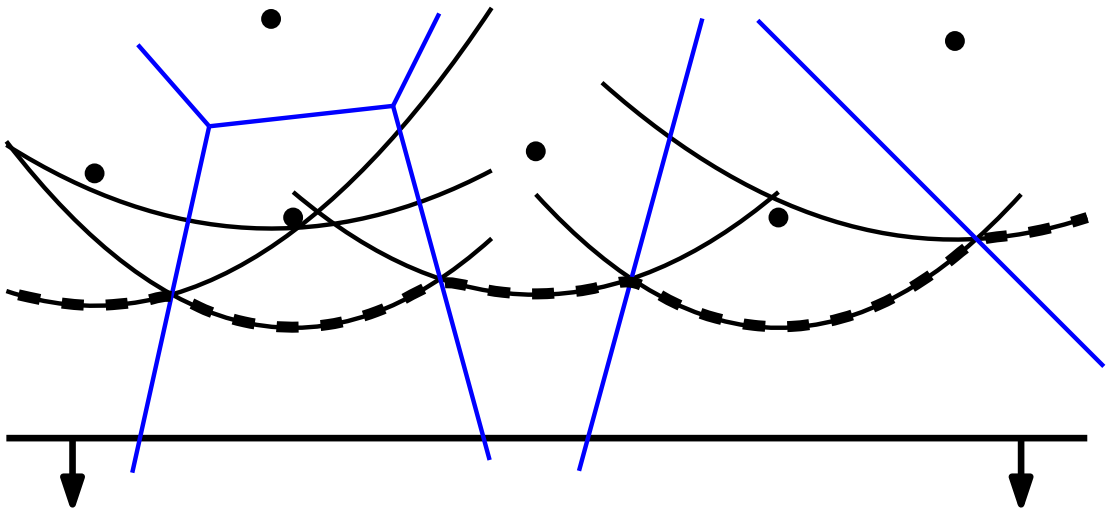
Plane sweep: Note that the Voronoi diagram above the sweep line may be affected by sites below the sweep line

Maintain and grow the portion of Voronoi diagram above the sweep line that is *known for sure*



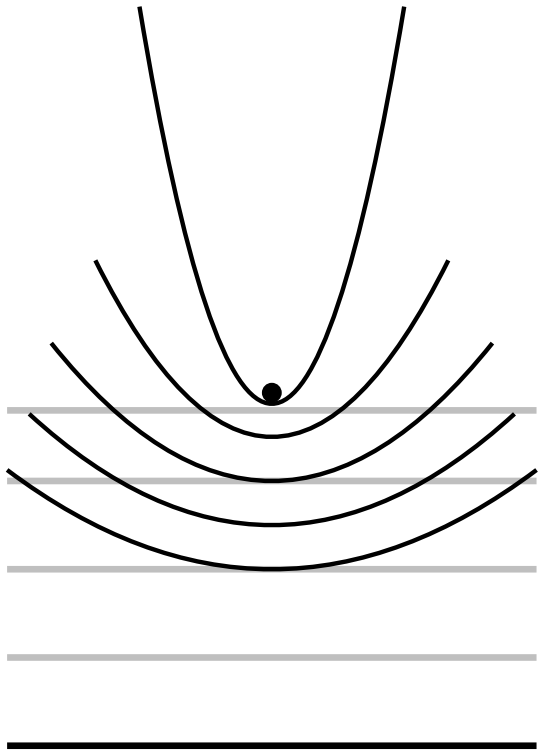
Beach line

The beach line separates the known and unknown part of the Voronoi diagram, it is the minimum of the parabolas defined by sites above the sweep-line and the sweep-line itself



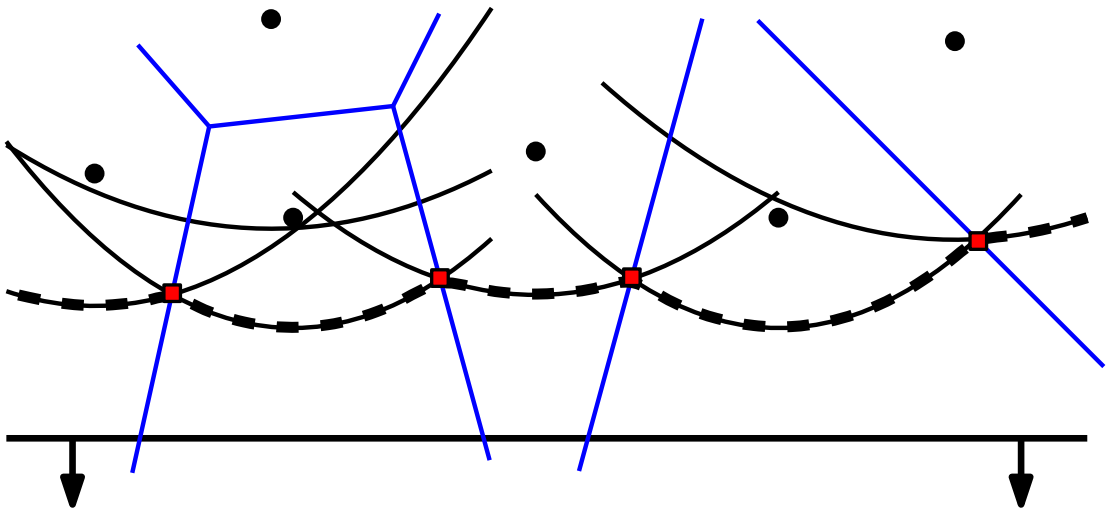
Beach line

The beach line changes continuously, even one parabola does



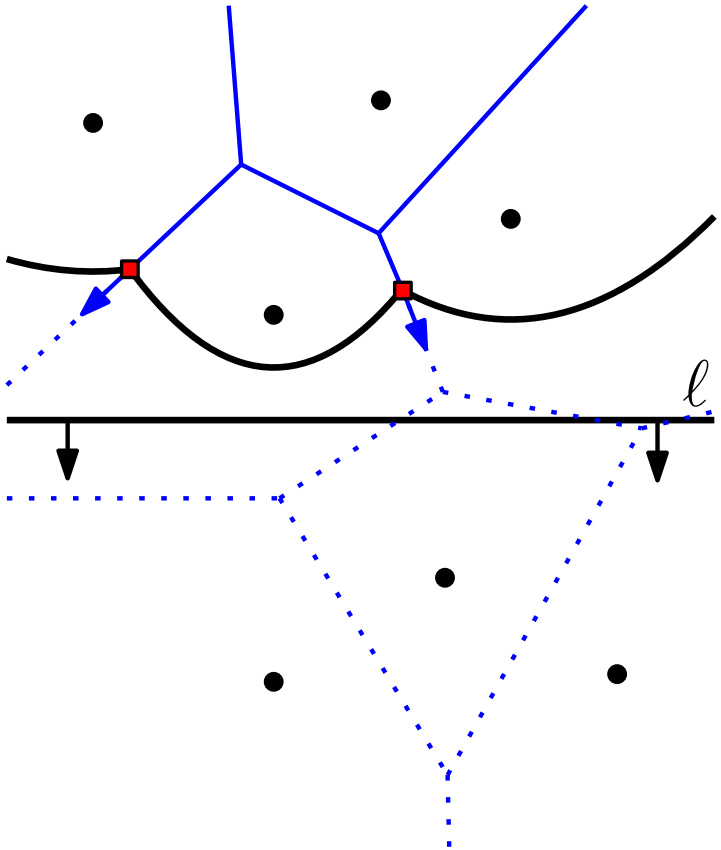
Beach line

Question: The beach line has break points, what do they represent?



Beach line

The break points move *and trace out the Voronoi diagram edges*



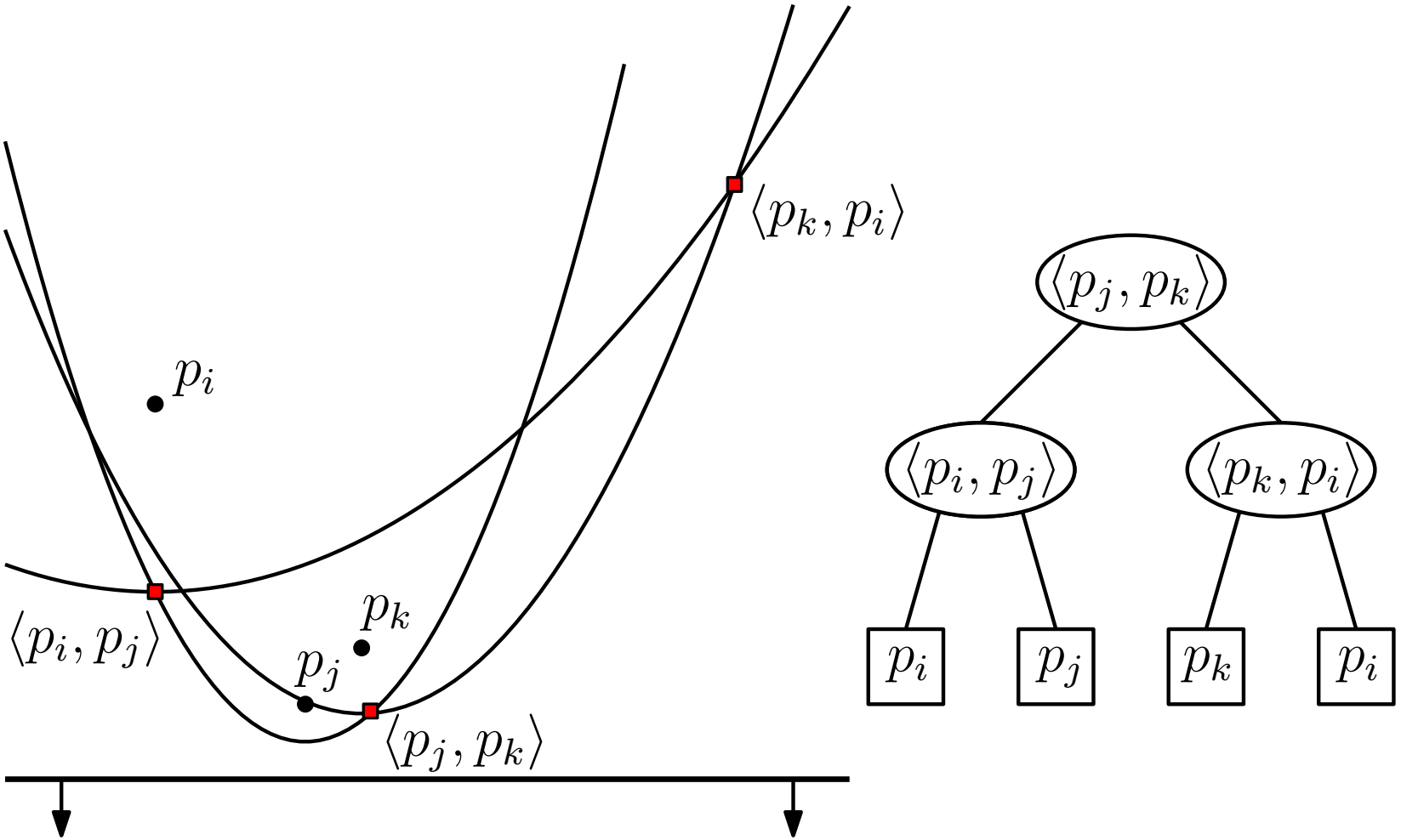
Status

Status: The ordered sequence of parabolic arcs that define the beach line; each is defined by a site (and the sweep-line)

Break points are defined by two sites (and the sweep-line)

Since the beach line is x-monotone, we can store the status in a balanced binary search tree on x-coordinate

Status


$\langle p_k, p_i \rangle$
p_i
$\langle p_i, p_j \rangle$
p_k
p_j
$\langle p_j, p_k \rangle$
$\langle p_j, p_k \rangle$
$\langle p_i, p_j \rangle$
$\langle p_k, p_i \rangle$
p_i
p_j
p_k
p_i

Other data structures

The sweep algorithm also needs an **event list** and a data structure to store the Voronoi diagram computed so far

The Voronoi diagram will be computed inside a large bounding box so that a doubly-connected edge list can be used

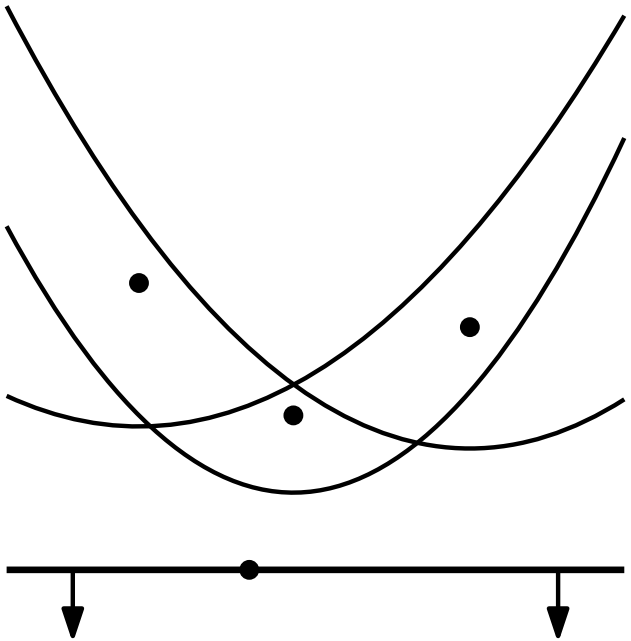
Events

The events are where the status changes = where the beach line changes

- When the sweep-line reaches a new site
- When a break point reaches the end of the edge it traces

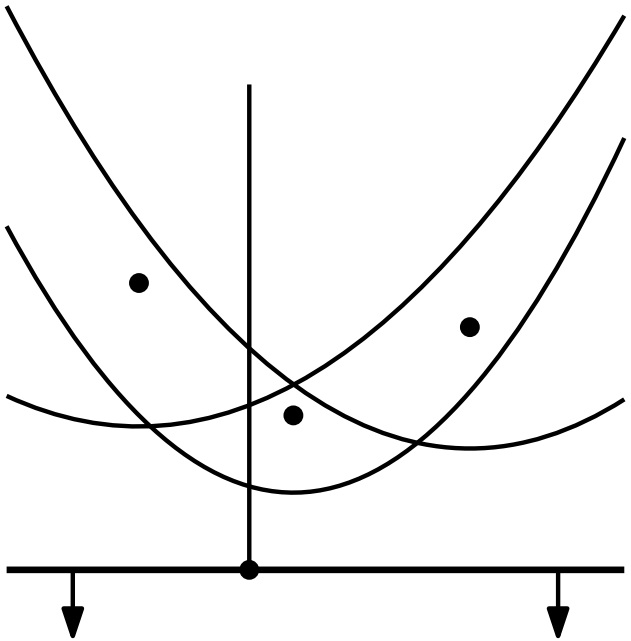
Site events

The sweep-line reaches a new site, a site event: a new parabola starts



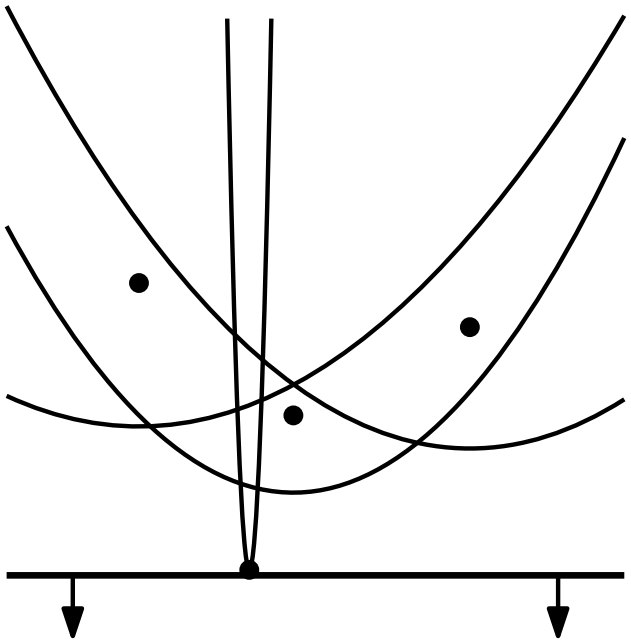
Site events

The sweep-line reaches a new site, a site event: a new parabola starts



Site events

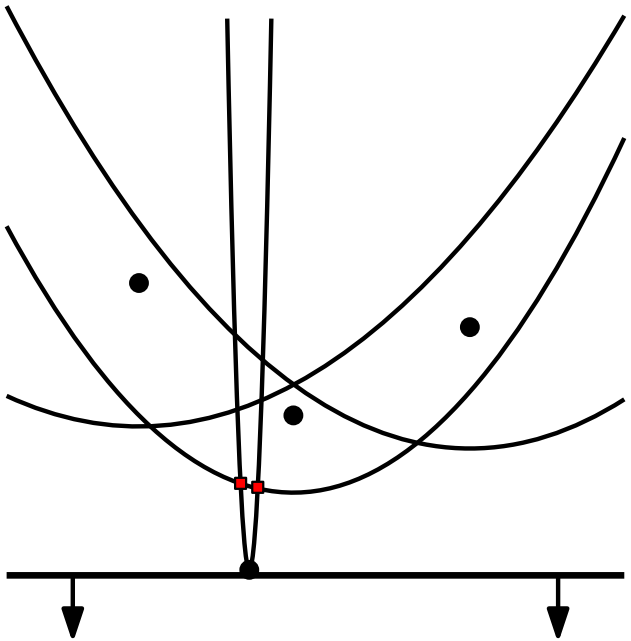
The sweep-line reaches a new site, a site event: a new parabola starts



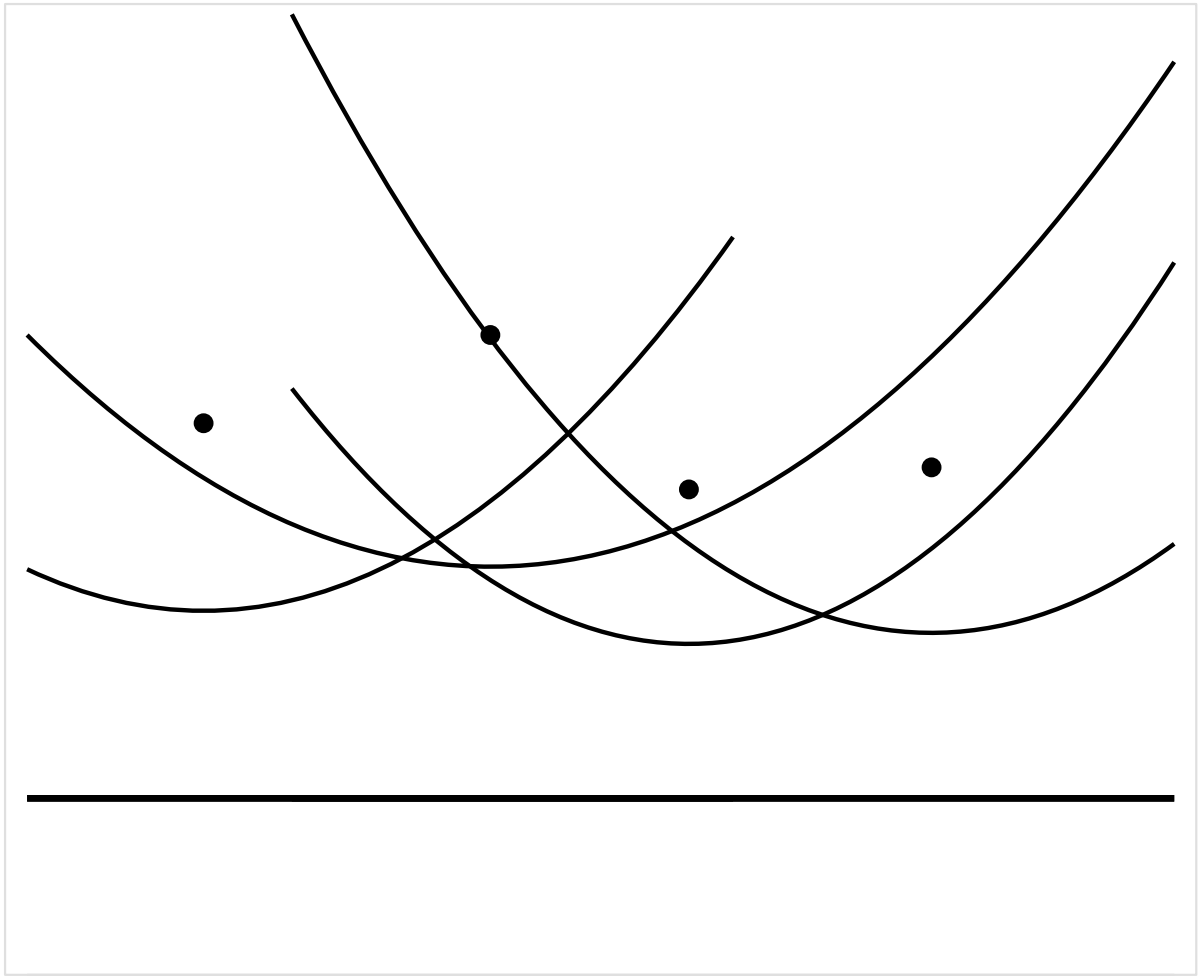
Site events

The sweep-line reaches a new site, a site event: a new parabola starts

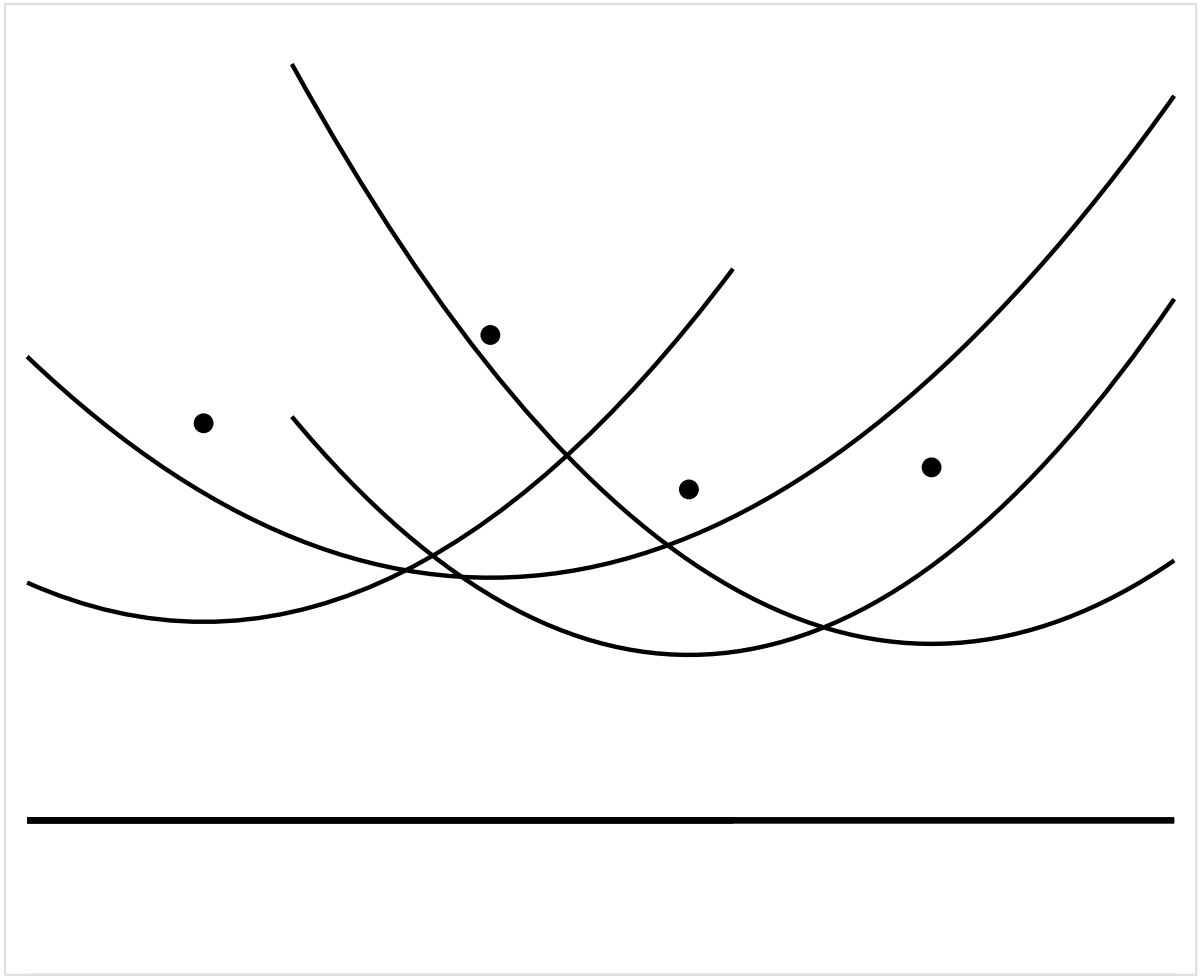
- Two new break points appear on the beach line
- A new Voronoi edge is discovered



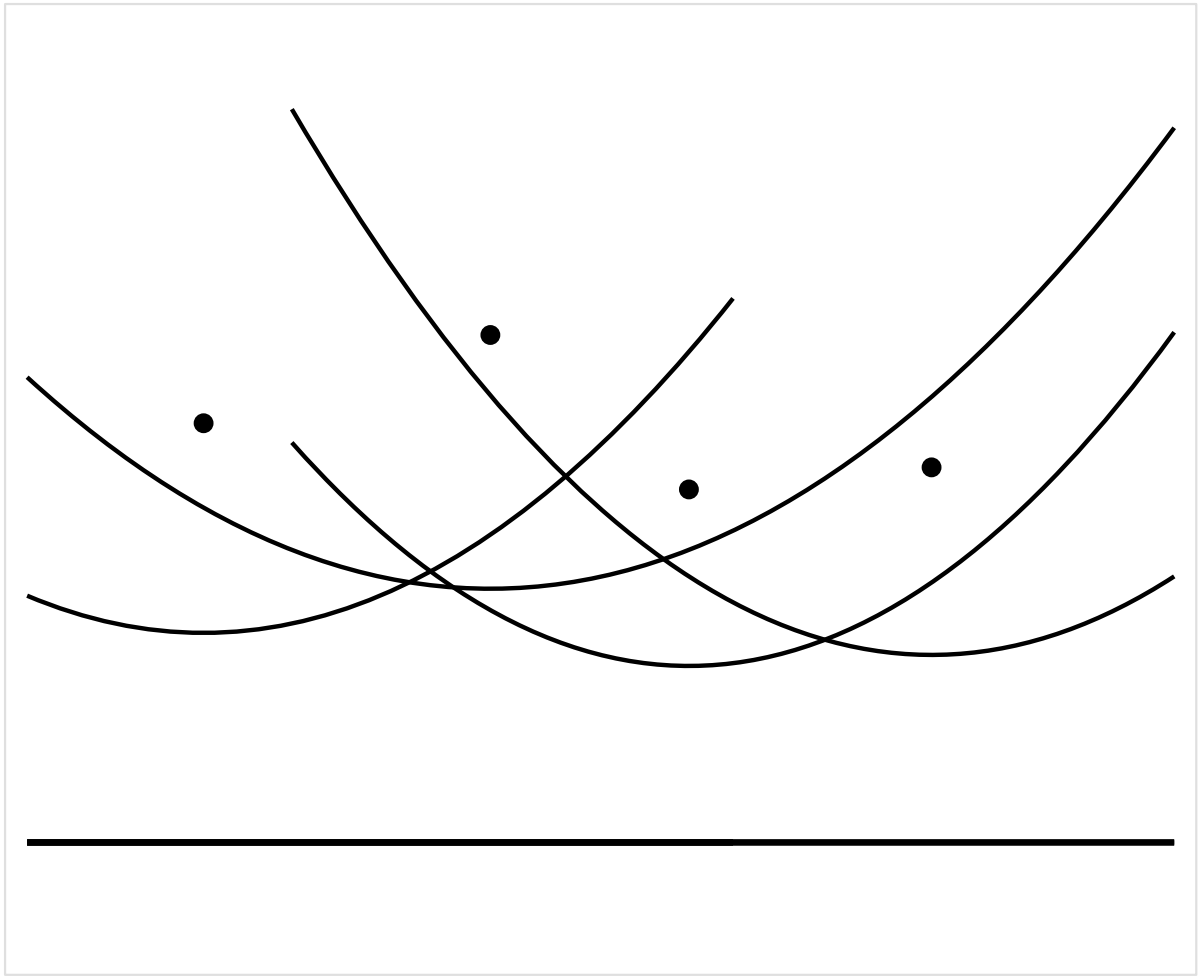
The other events



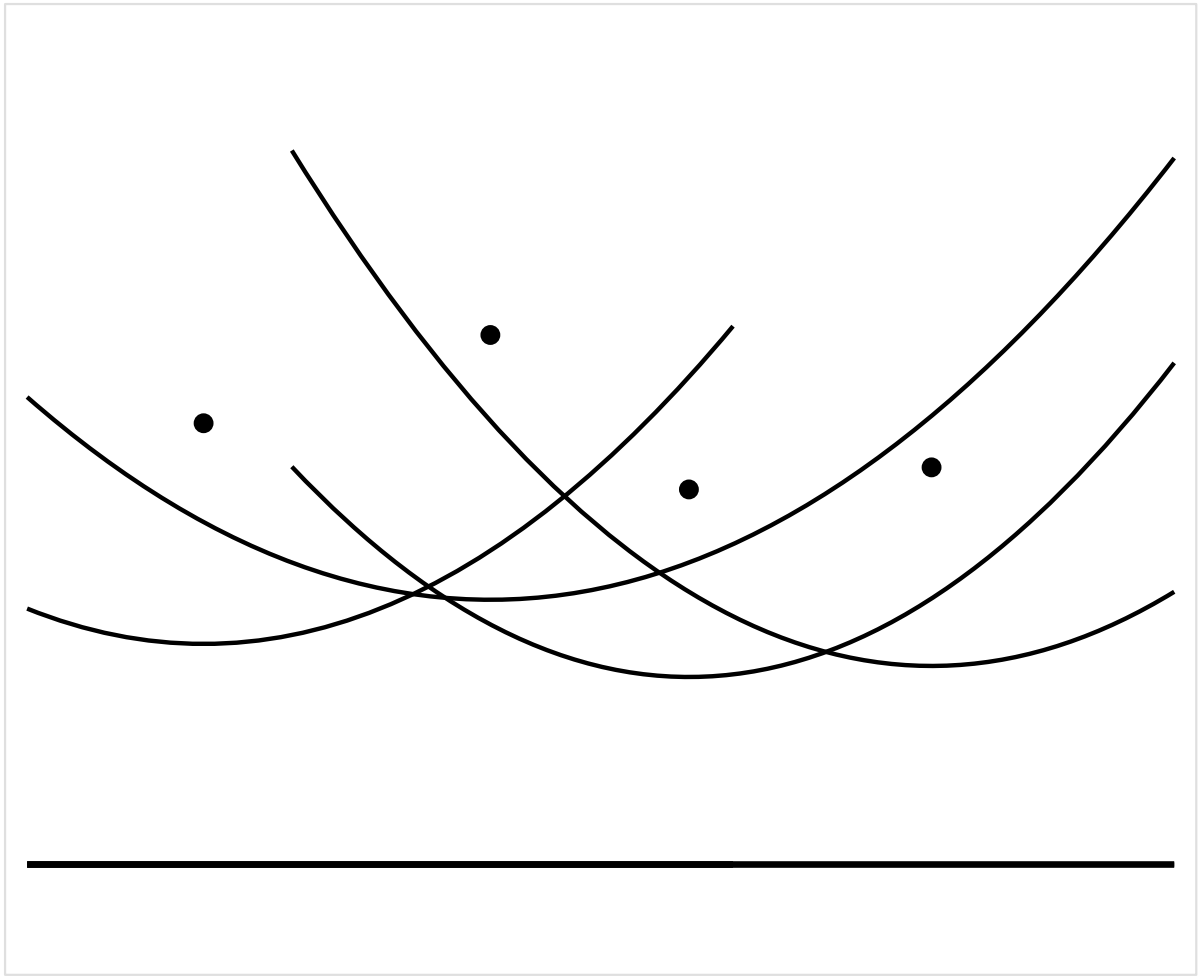
The other events



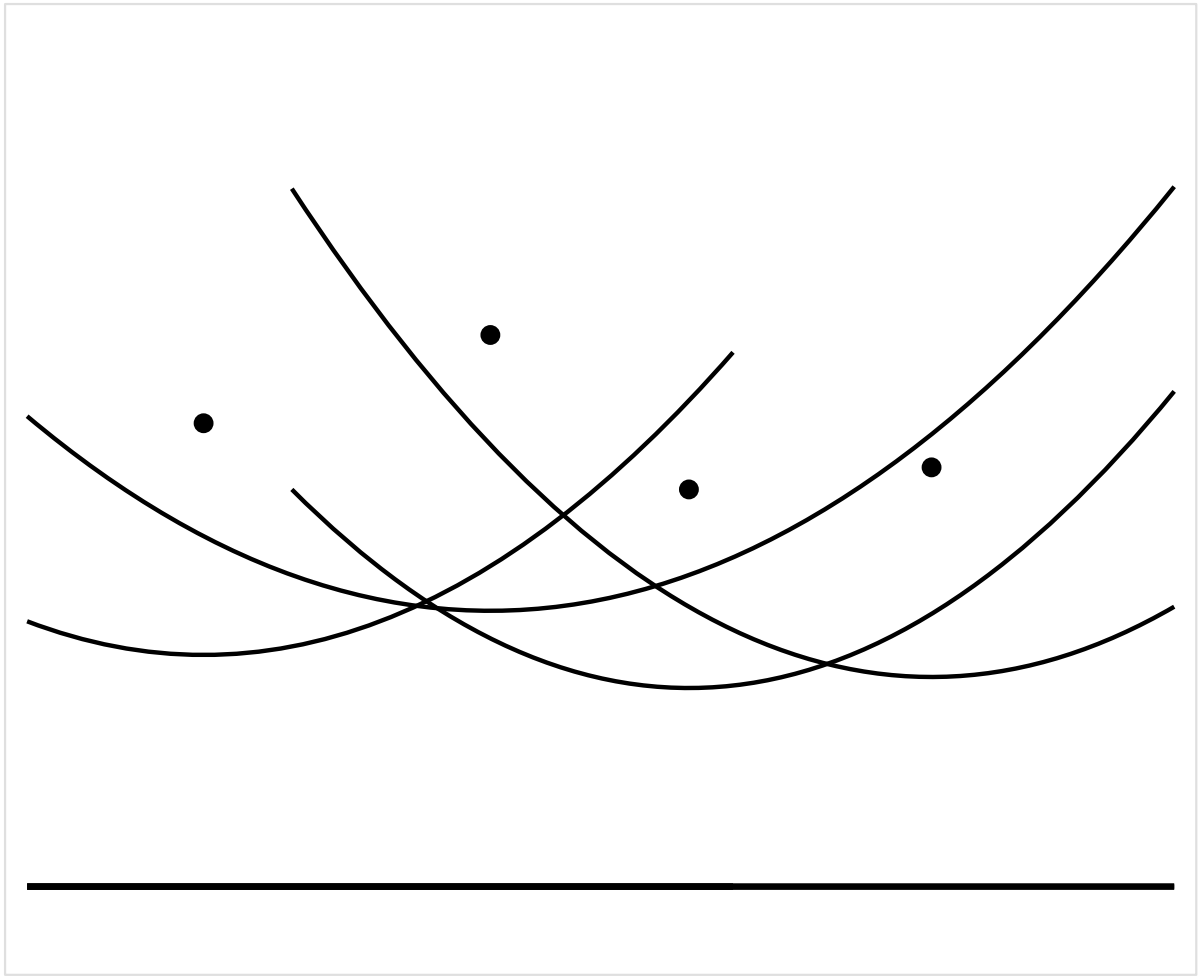
The other events



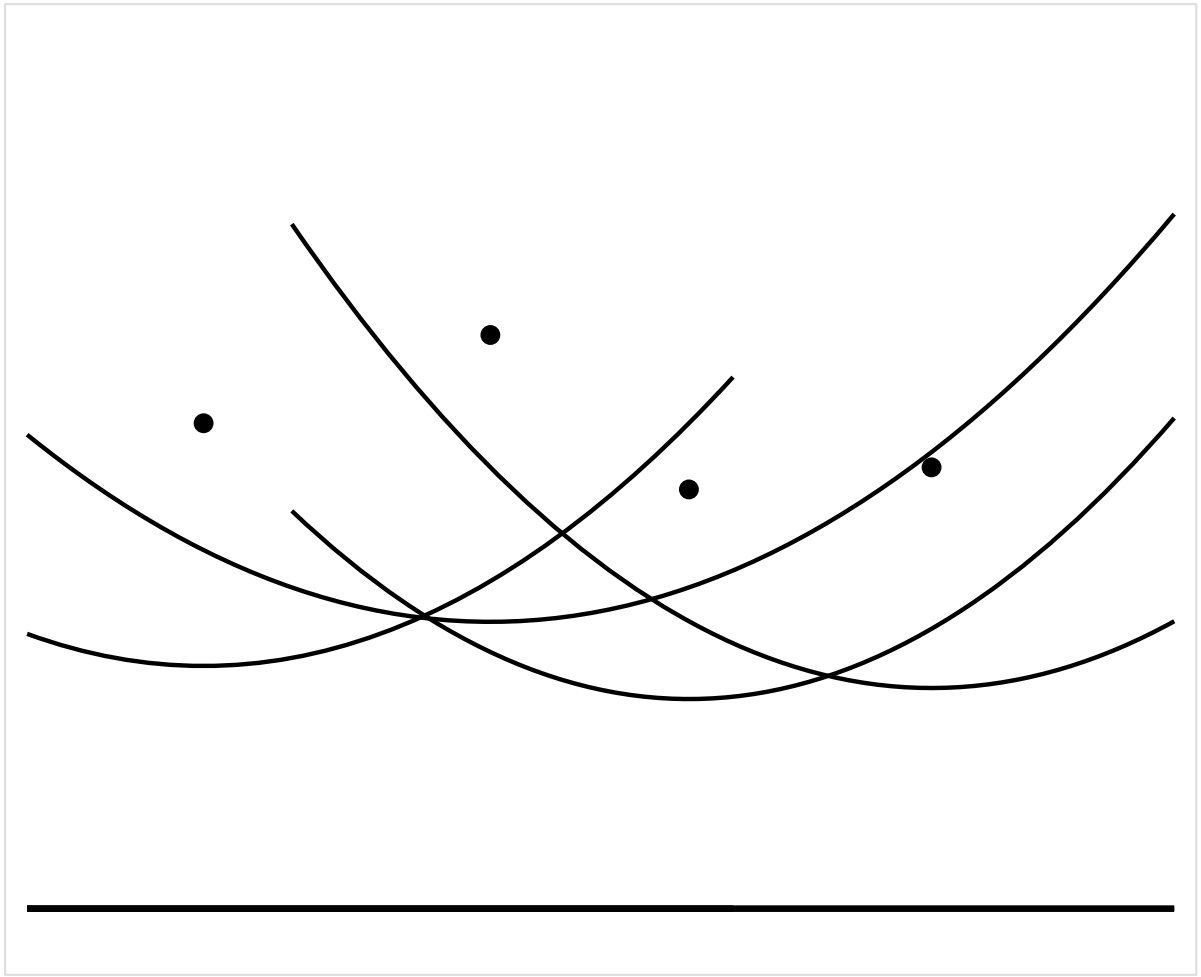
The other events



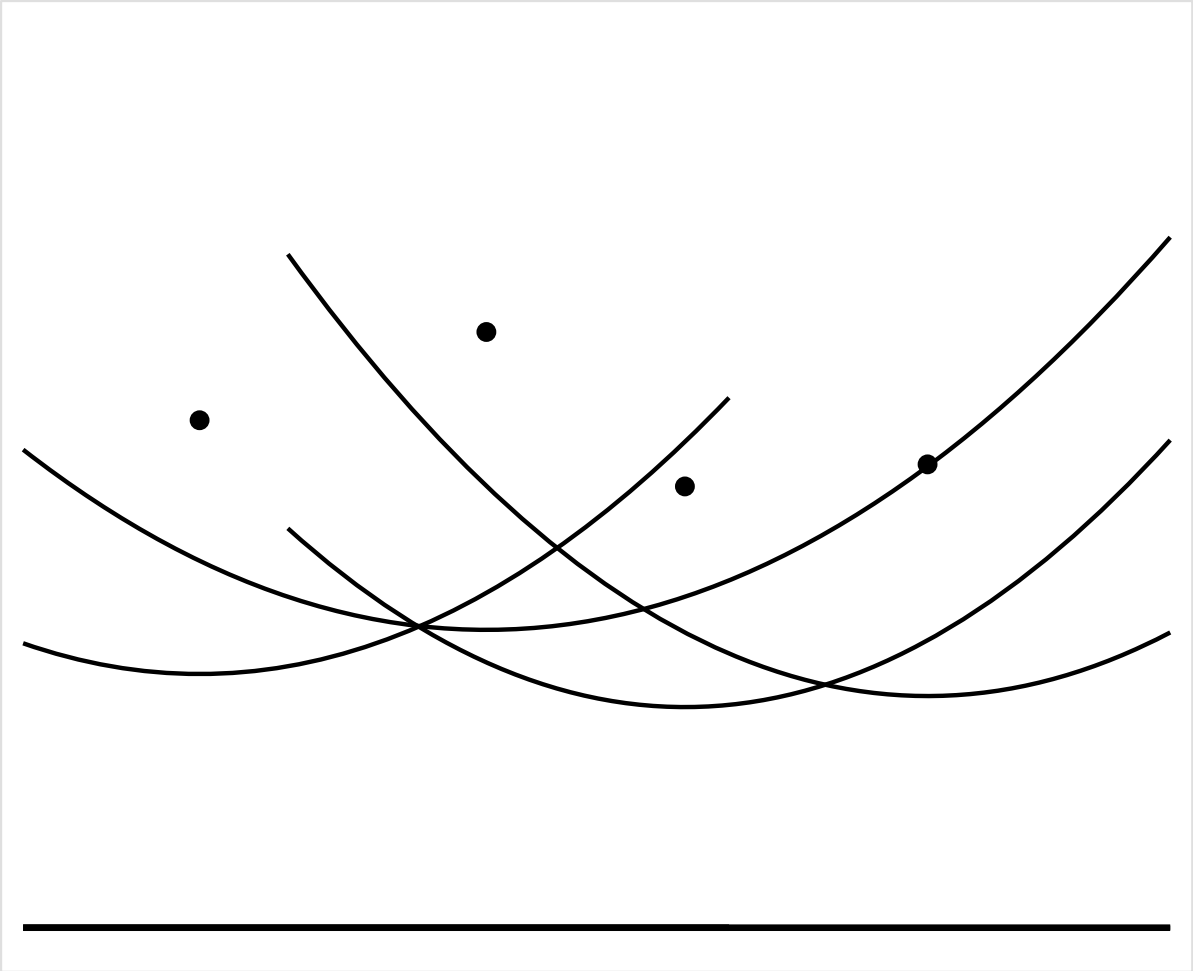
The other events



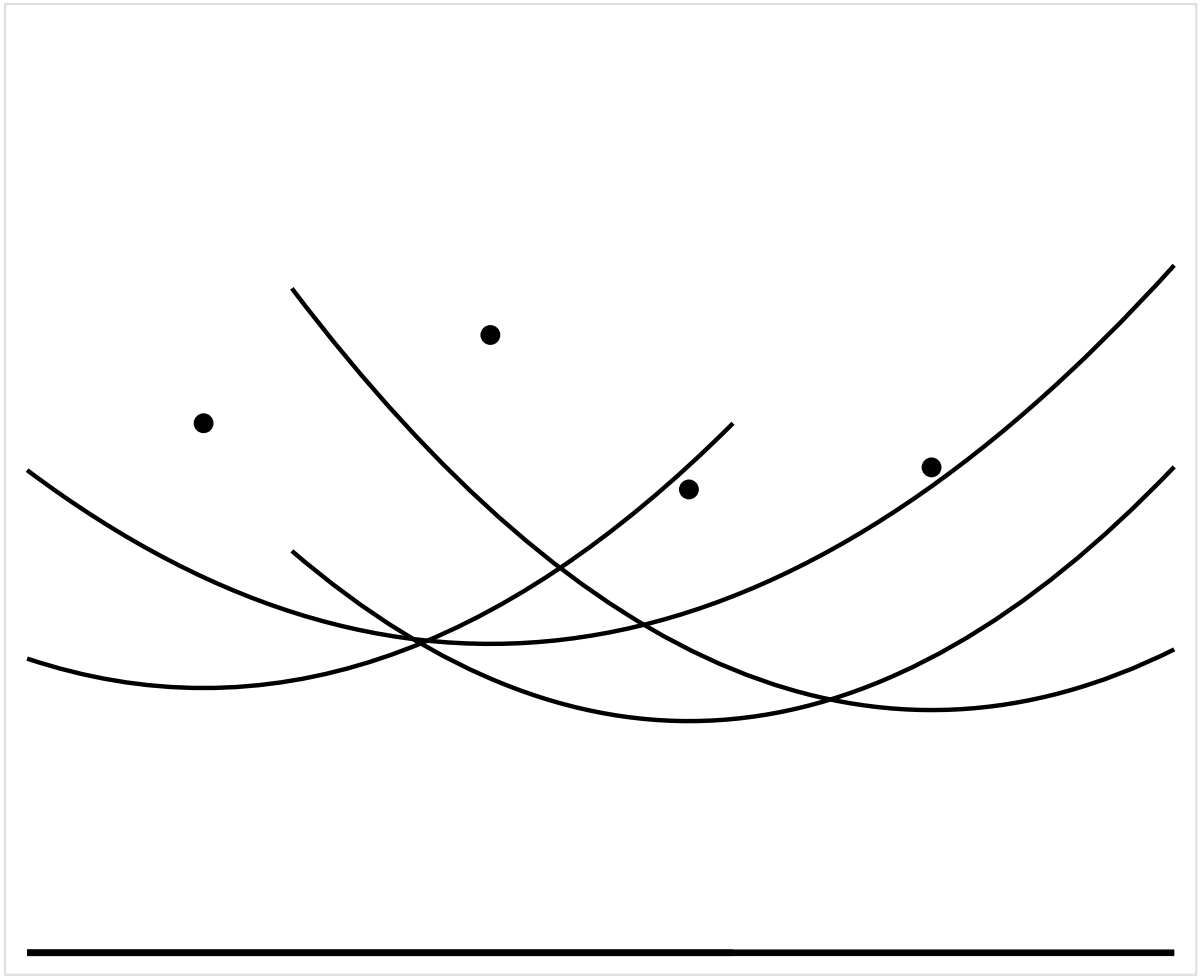
The other events



The other events

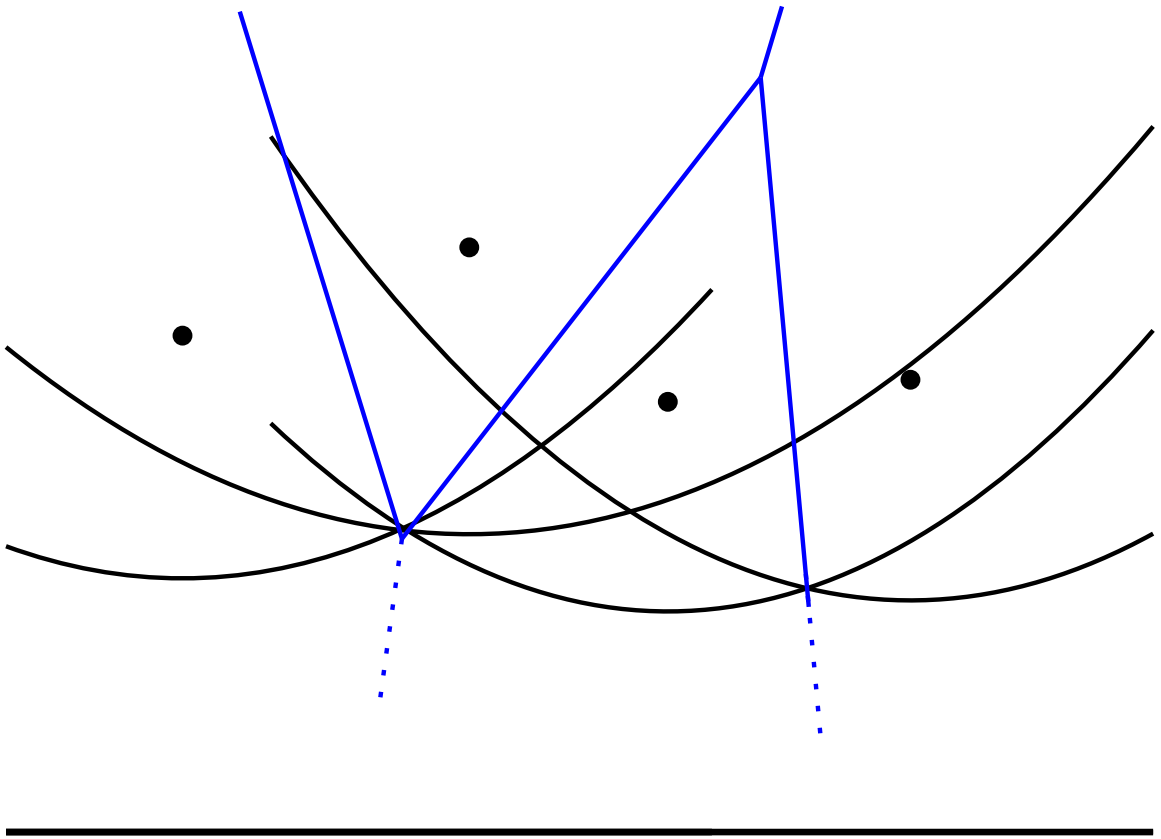


The other events



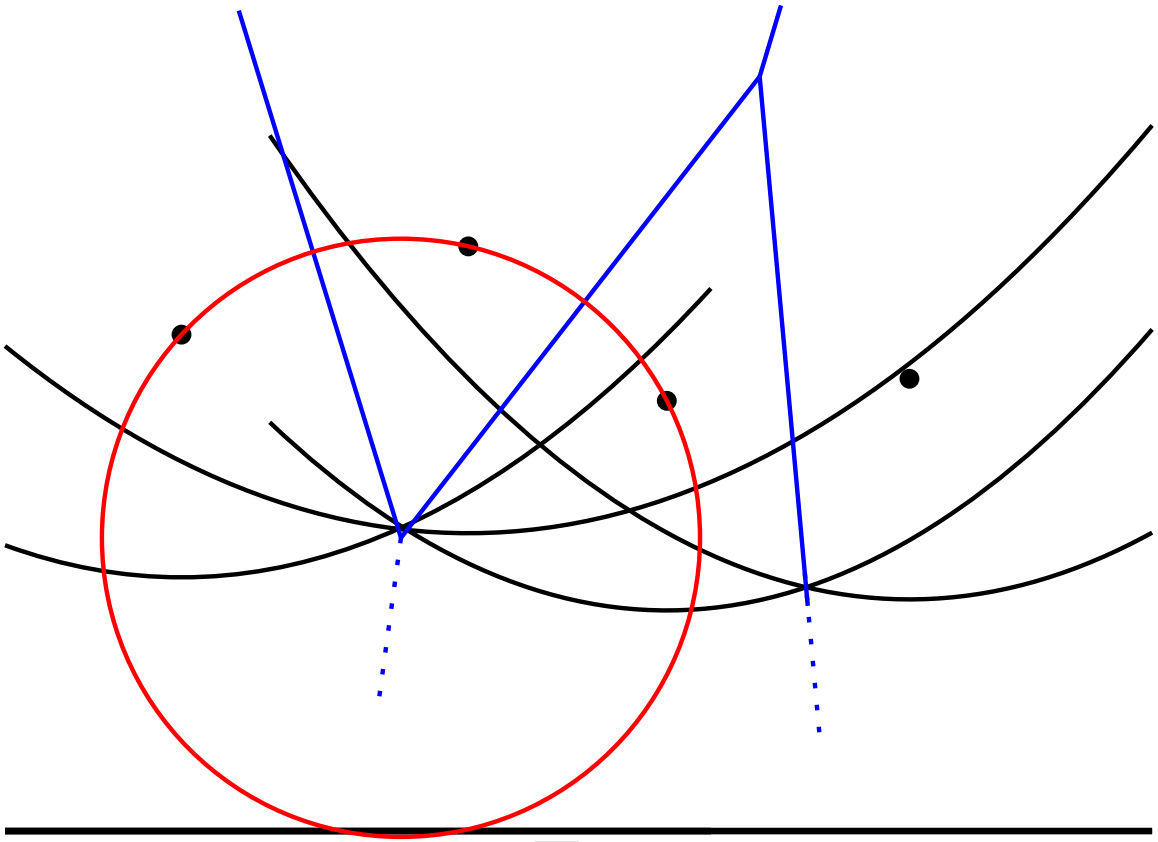
The other events

Parabolic arcs may disappear from the beach line



The other events

We discover an empty circle and a Voronoi vertex

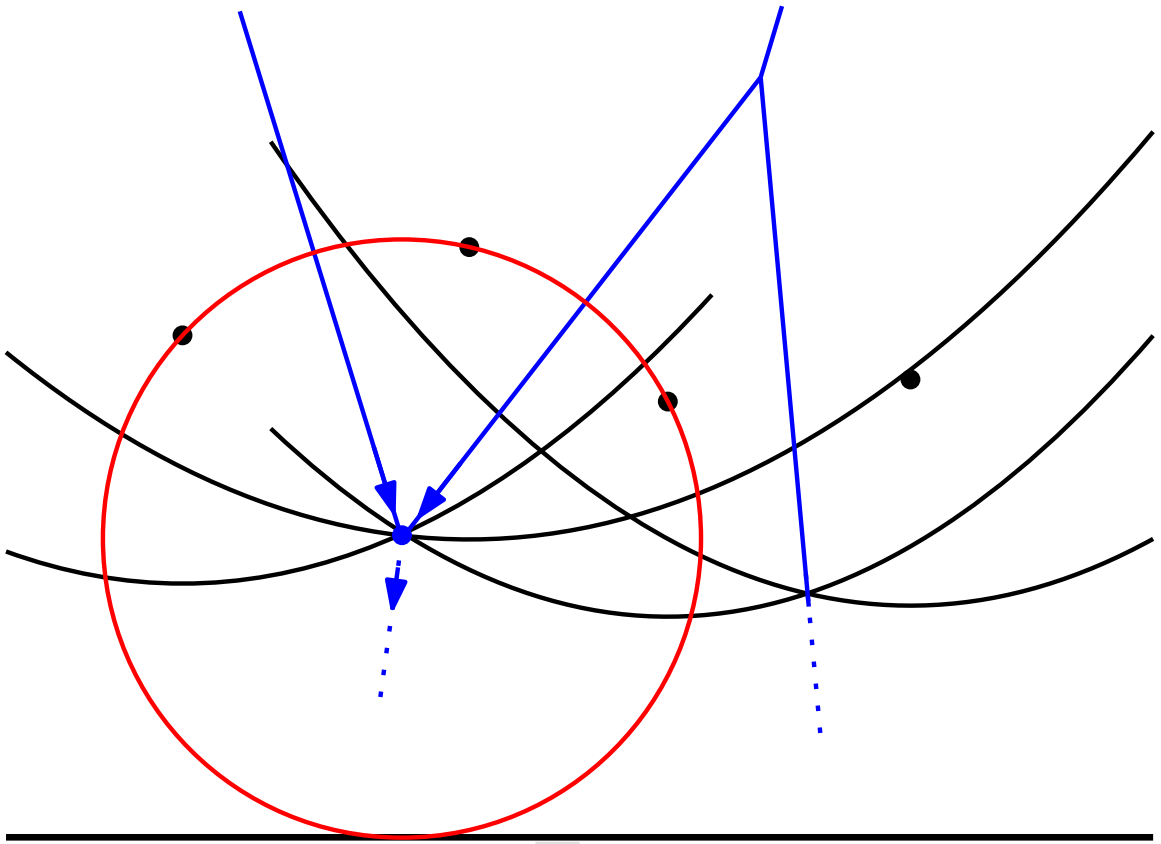


Circle events

At a circle event:

- A parabolic arc disappears from the beach line
- Two adjacent break points come together
- A Voronoi vertex is discovered as the vertex incident to two known Voronoi edges
- A new break point starts to be traced
- The sweep line reached the bottom of an empty circle through 3 sites

Circle events



Circle events can only happen for three sites that have adjacent parabolic arcs on the beach line

Site and circle events

The only way for a new parabolic arc to **appear** on the beach line is through a site event

The only way for a parabolic arc to **disappear** from the beach line is through a circle event

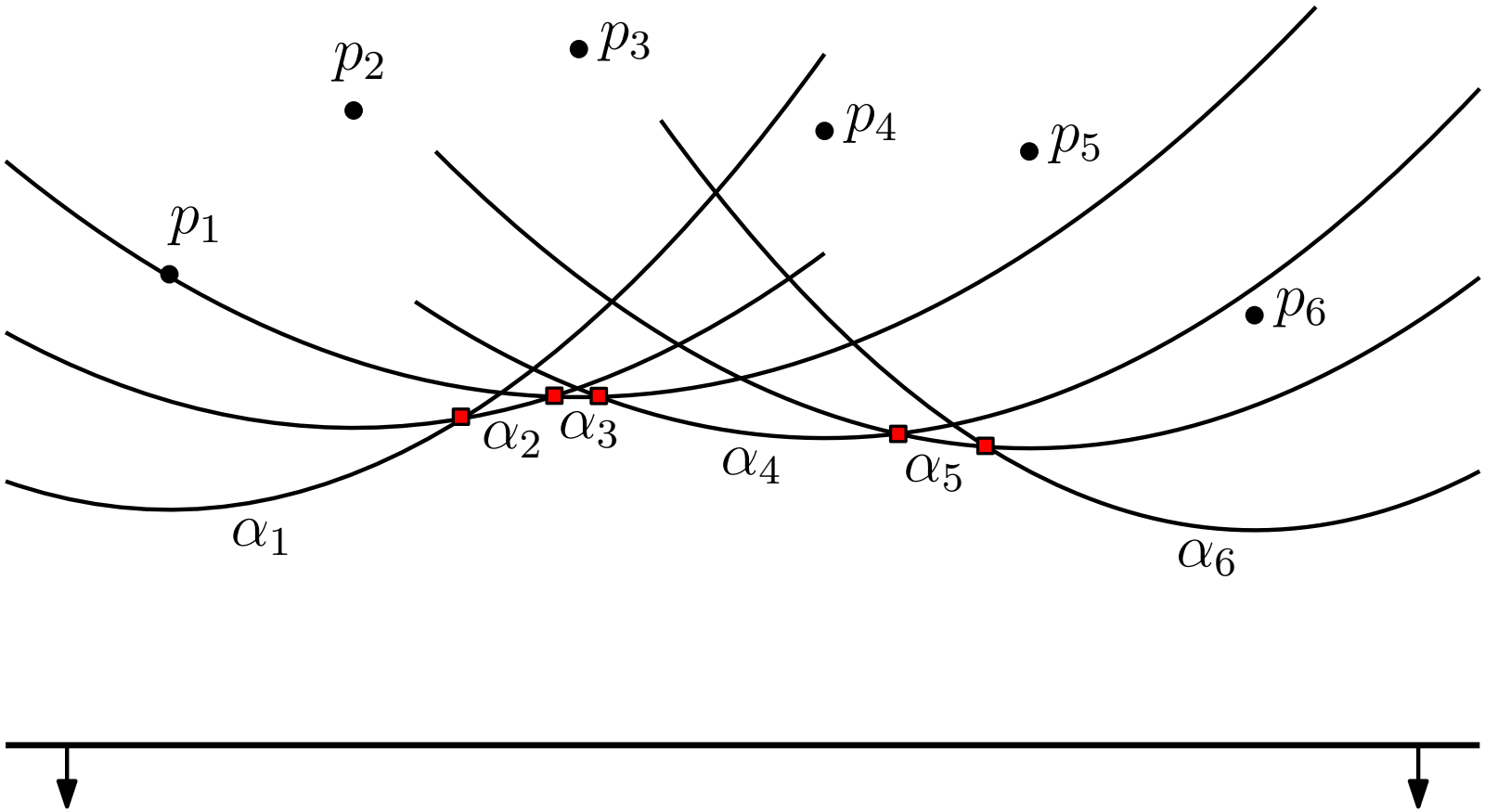
There are no other events

Site and circle events

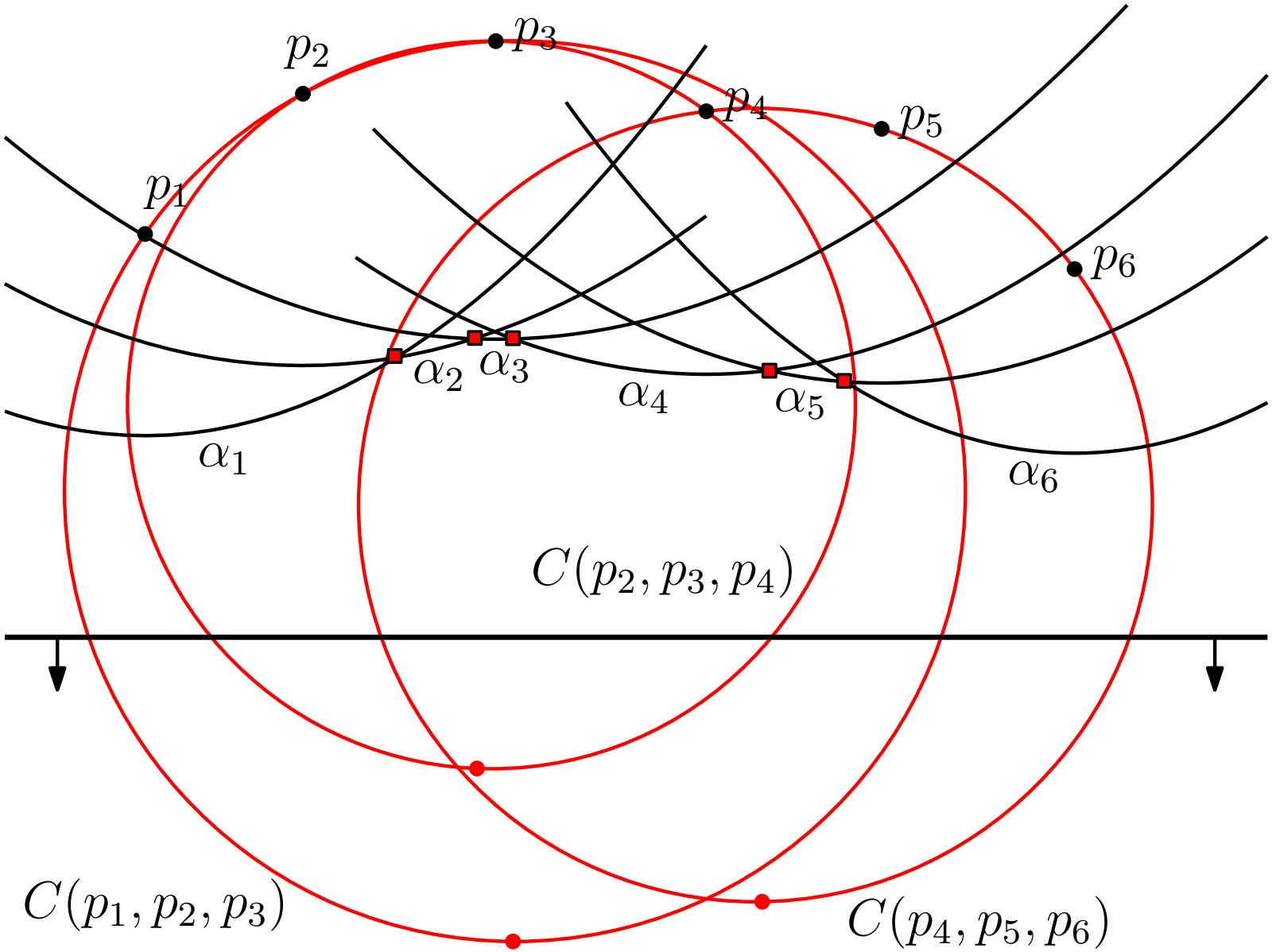
There are n site events and they are known in advance

Question: How can we know circle events before they occur?

Detecting circle events



Detecting circle events



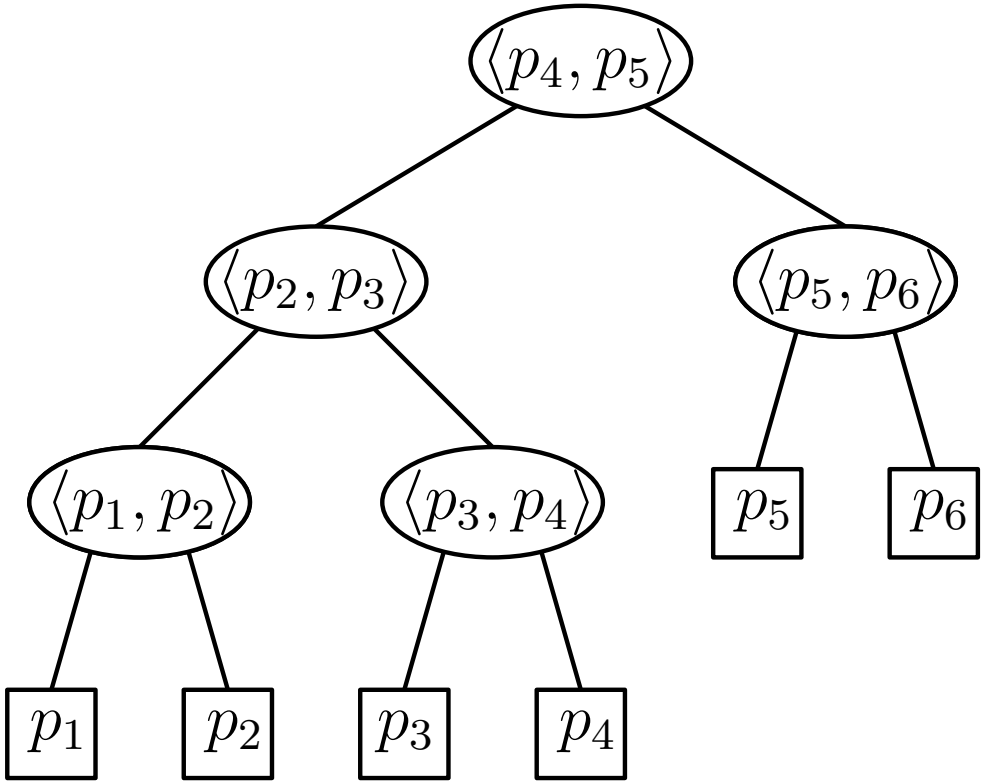
Detecting circle events

A circle event occurs if the sweep line reaches the bottom of an empty circle defined by three sites that have consecutive parabolic arcs on the beach line

We will make sure that any three sites that have consecutive arcs on the beach line and whose circle has its lowest point below the sweep line have this lowest point as circle event in the event list

Detecting circle events

In the status structure we can see all triples of consecutive parabolic arcs that can give circle events



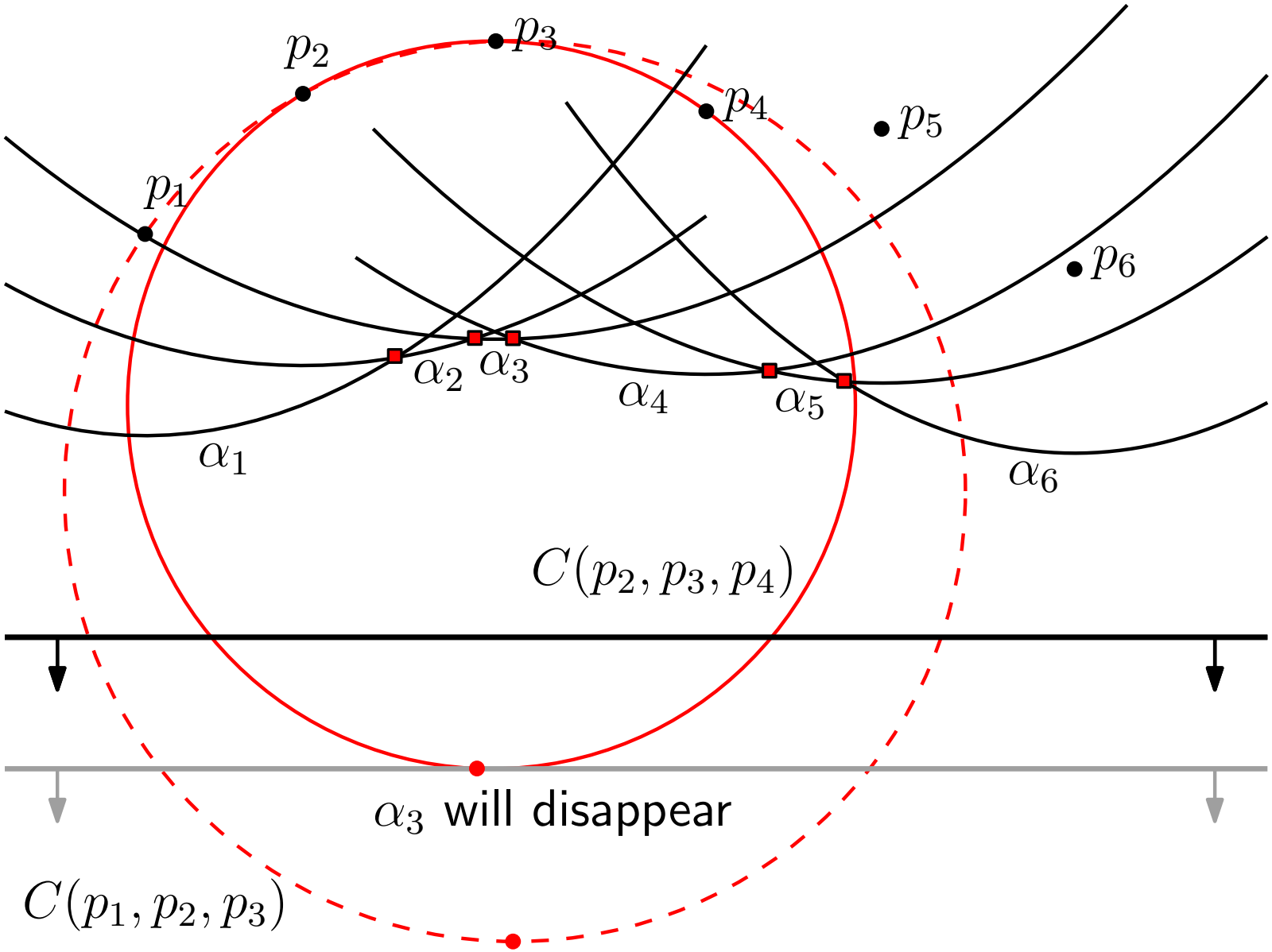
False alarms

We may have stored a circle event in the event list, but it may be that it never happens ...

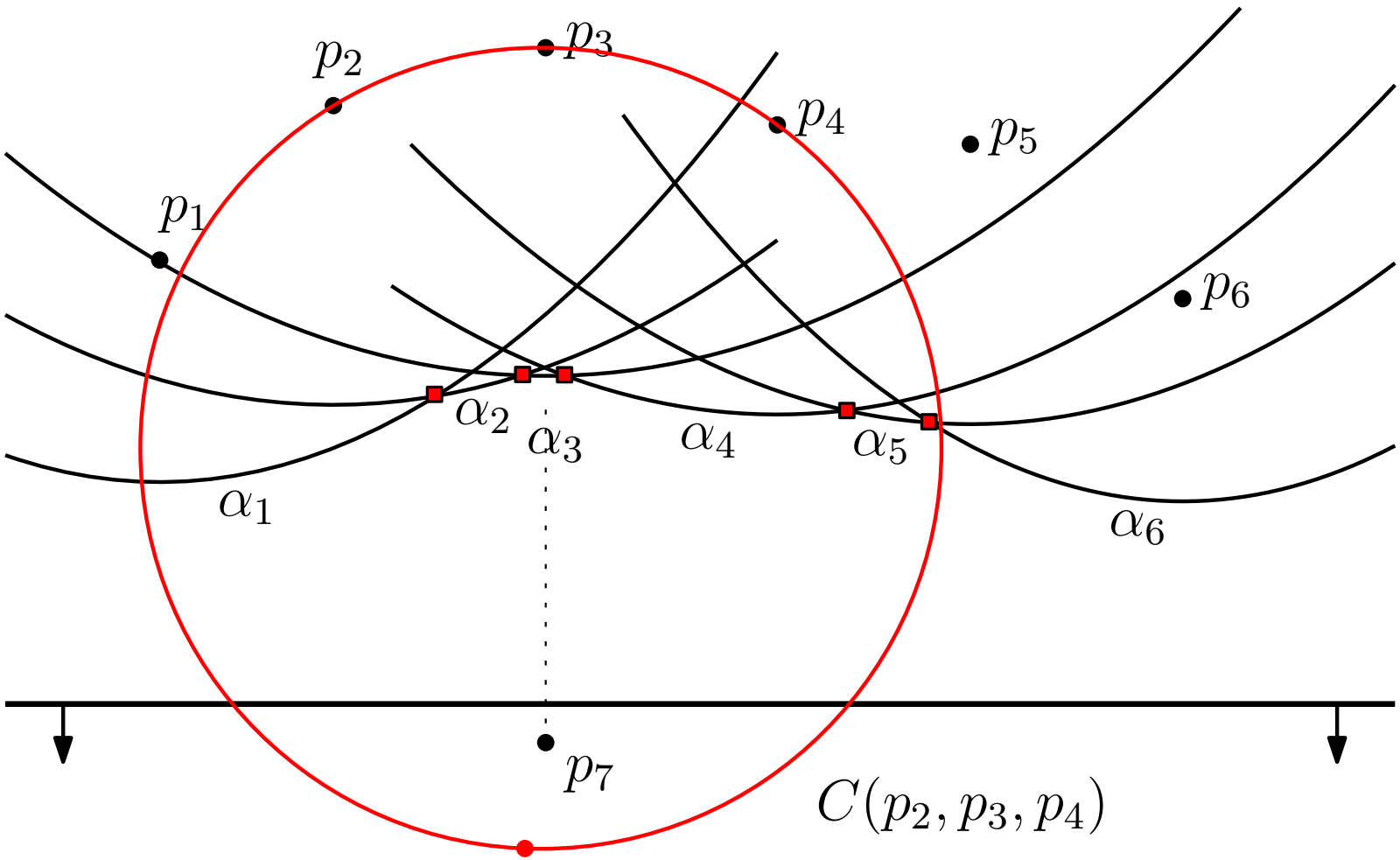
This is called a false alarm

There are two reasons for false alarms: site events and other circle events

False alarms



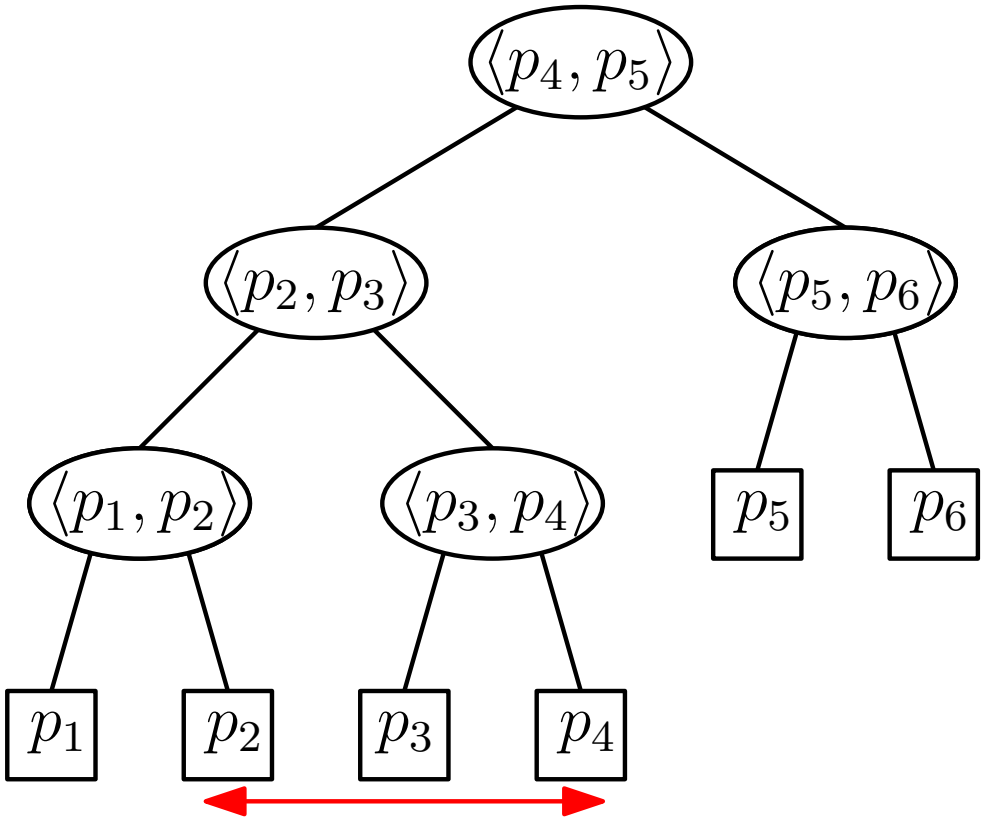
False alarms



The circle of a circle event may turn out not to be empty

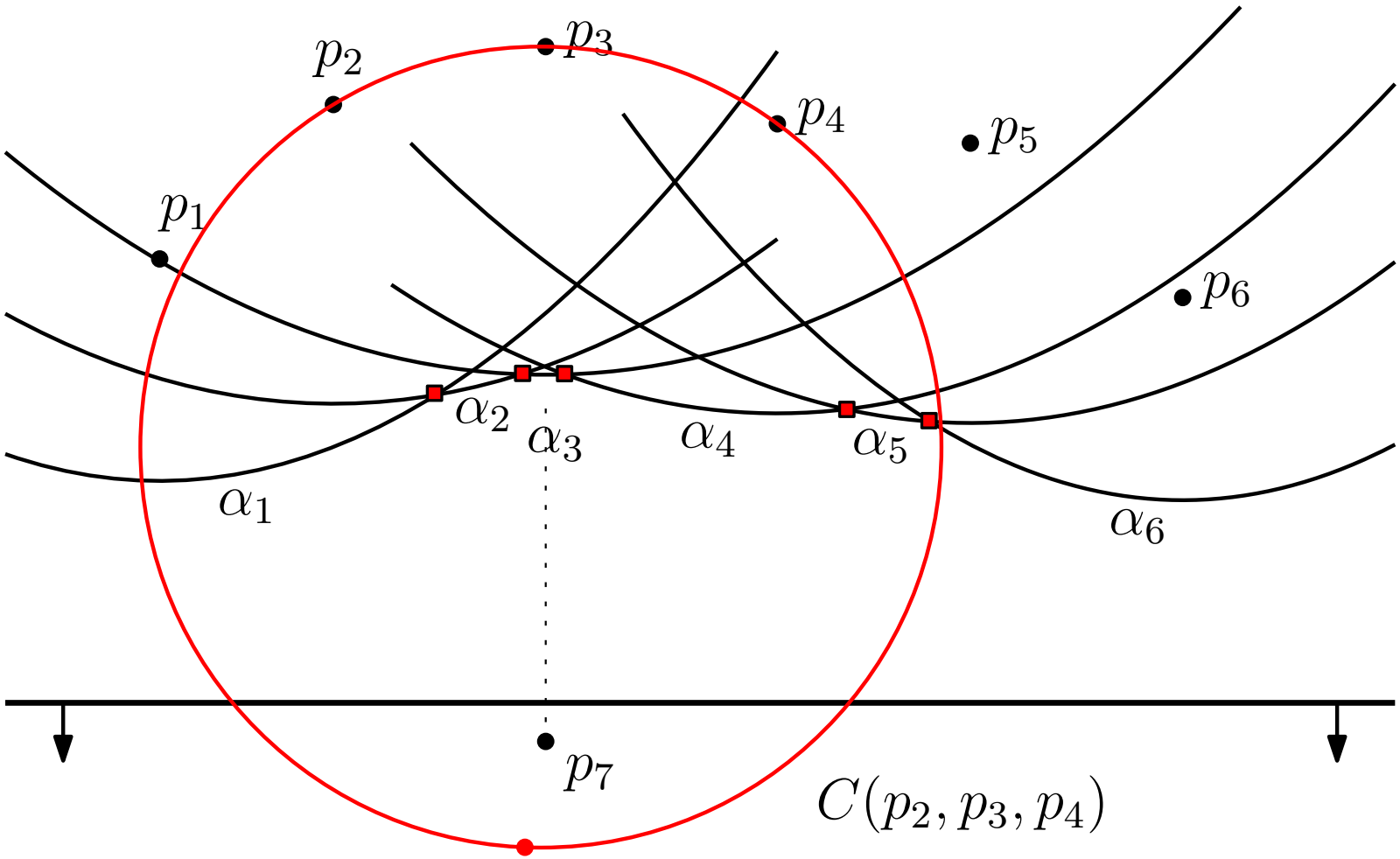
Detecting false alarms

A site event that disrupts three consecutive parabolic arcs



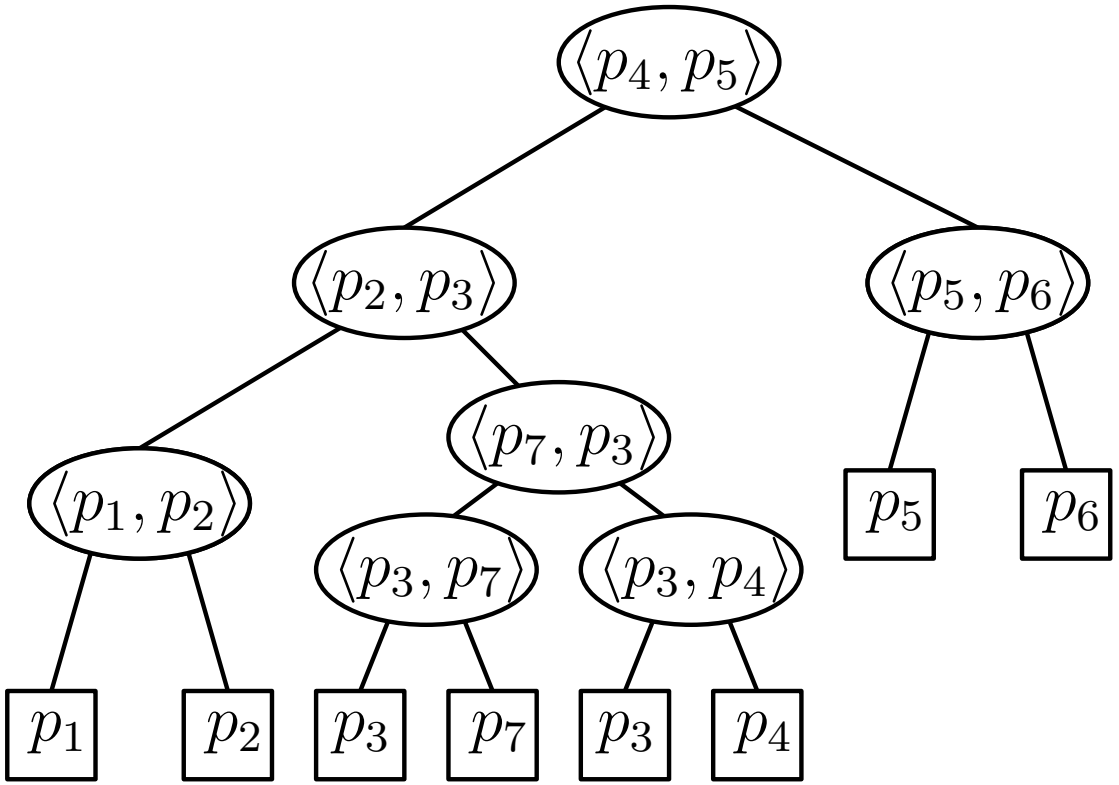
circle event $C(p_2, p_3, p_4)$

Detecting false alarms



Detecting false alarms

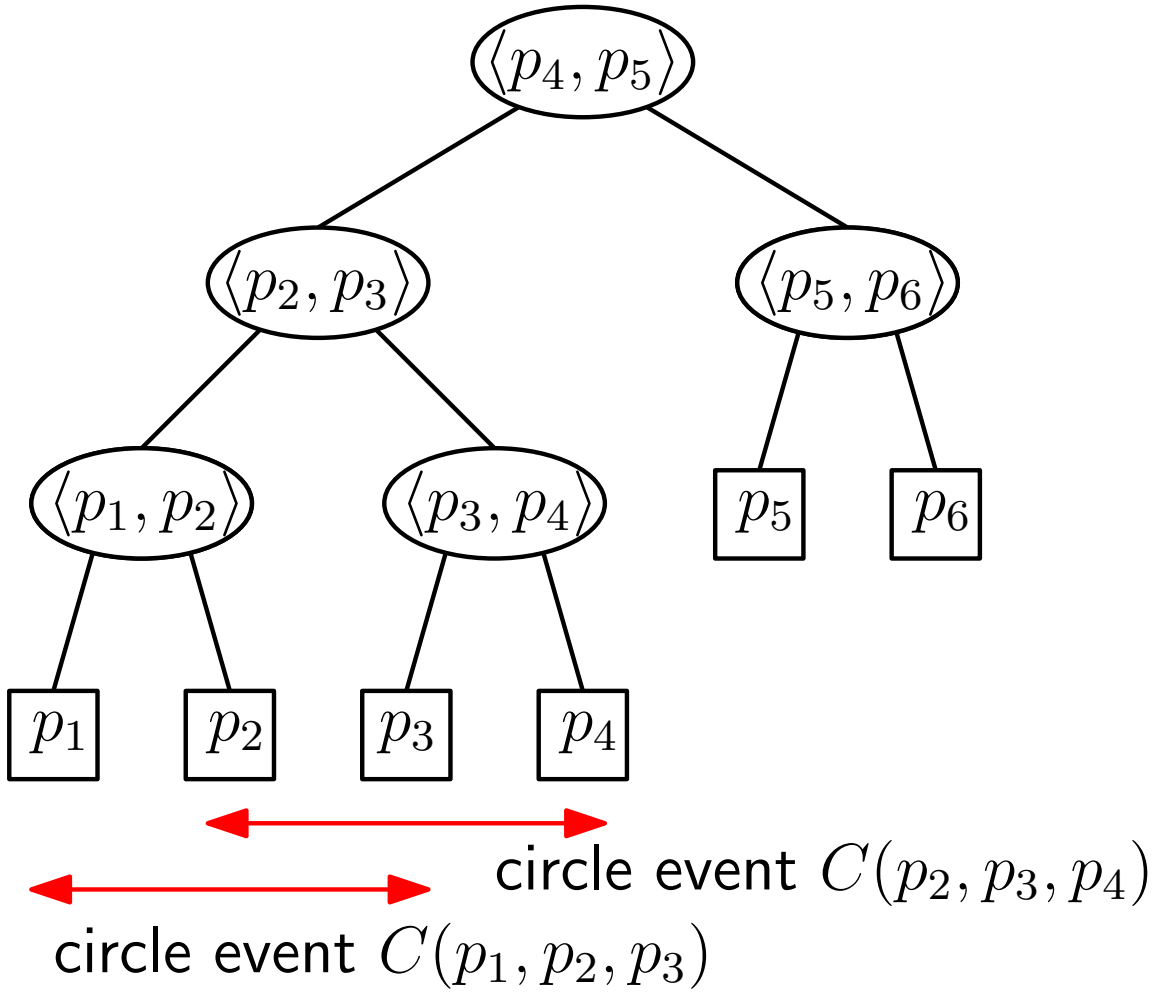
A site event that disrupts three consecutive parabolic arcs



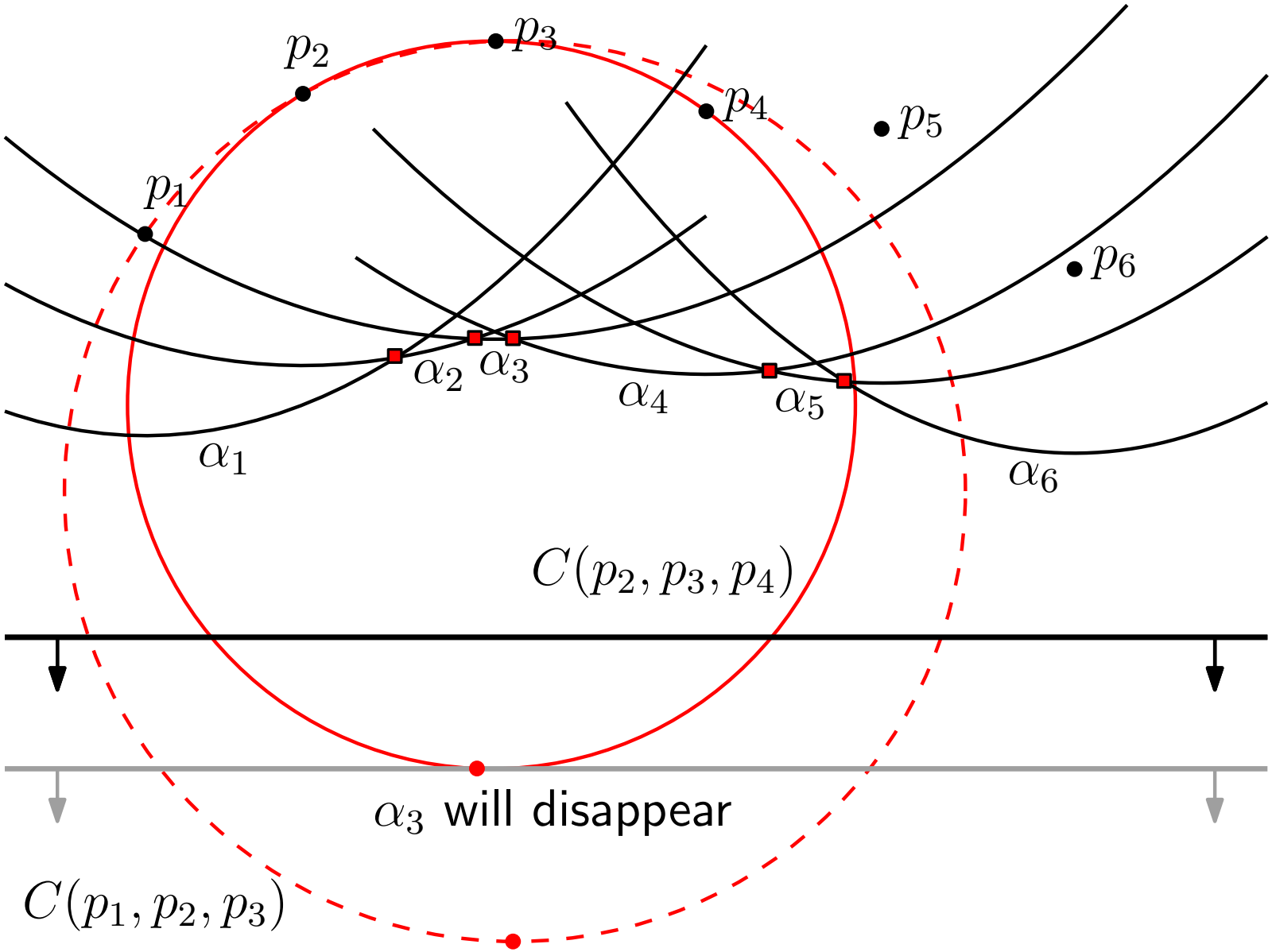
circle event $C(p_2, p_3, p_4)$ gone

Detecting false alarms

A circle event that disrupts three consecutive parabolic arcs

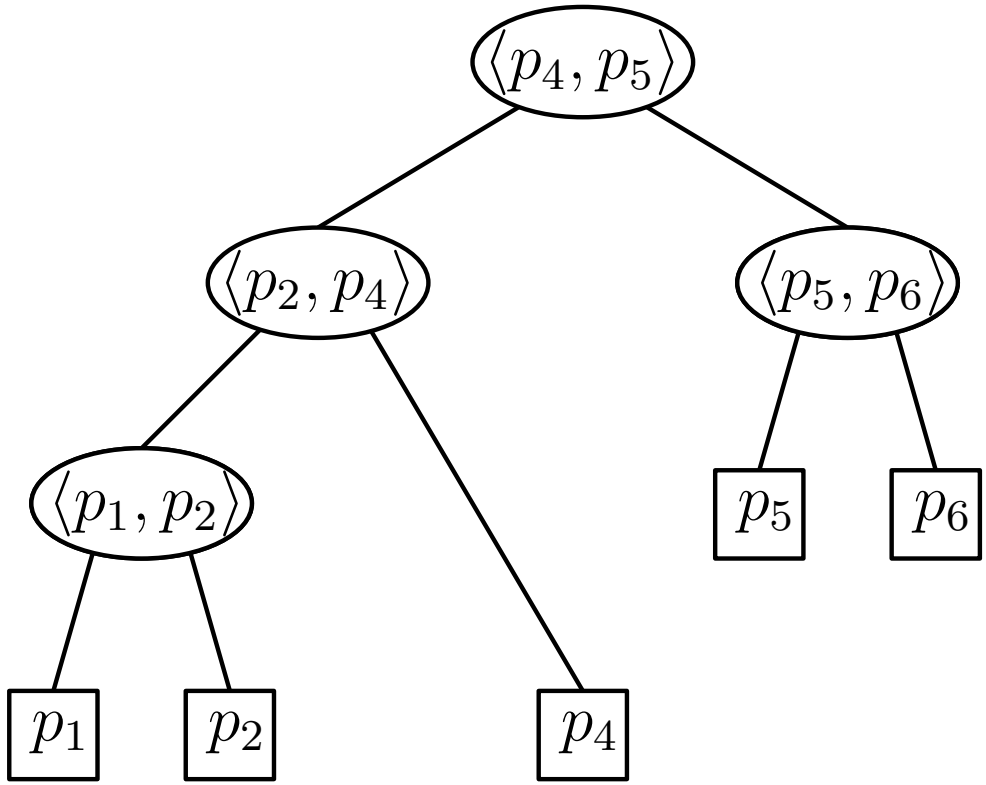


Detecting false alarms



Detecting false alarms

A circle event that disrupts three consecutive parabolic arcs



circle event $C(p_1, p_2, p_3)$ gone

The data structures

Recall we have a status structure $\mathcal{T}$, an event list, and a DCEL

We need pointers from $\mathcal{T}$ into the DCEL to be able to update it efficiently

We need pointers from the leaves of $\mathcal{T}$ into the event list to be able to remove circle events if they are false alarms

The data structures

A leaf of $\mathcal{T}$ has a pointer to all events in which the parabolic arc participates

Easy question: How many can there be, at most?

Global algorithm

Algorithm VoronoiDiagram(P)

1. Initialize the event queue $\mathcal{Q}$ with all site events, initialize an empty status structure $\mathcal{T}$ and an empty doubly-connected edge list $\mathcal{D}$
2. **while** $\mathcal{Q}$ is not empty
3. **do** remove the event with largest y-coordinate from $\mathcal{Q}$
4. **if** the event is a site event, occurring at site p_i
5. **then** HandleSiteEvent(p_i)
6. **else** HandleCircleEvent(γ), where γ is the leaf of $\mathcal{T}$ representing the arc that will disappear
7. When all events are handled, we must still fix the doubly-connected edge list with respect to the bounding box, and to add face information

Site event actions

At a site event, at a site p_i on the sweep line ℓ:

- Find the parabolic arc α vertically above p_i in $\mathcal{T}$
- Remove the false alarm with α in the middle from the event list (if it exists)
- Update $\mathcal{T}$: one arc is split and a new one for p_i appears in between, and break points are updated
- Make two new half-edges for the detected Voronoi edge in the DCEL
- Add new circle events for the new consecutive triples (if the circle has its lowest point below ℓ)

Circle event actions

At a circle event, for circle $C(p_i, p_j, p_k)$ whose bottom is on the sweep line ℓ, and γ is the leaf of $\mathcal{T}$ whose arc disappears:

- Remove the false alarms that involve the parabolic arc corresponding to γ
- Update $\mathcal{T}$: remove the leaf γ and update break points
- Make a new vertex object for the Voronoi vertex, two new half-edge objects, and connect six half-edges and the vertex in the DCEL
- Add new circle events for the new consecutive triples (if the circle has its lowest point below ℓ)

Analysis

Any event removes at most two false alarms and generates at most two new circle events

Any event is handled in $O(\log n)$ time

There are n site events and at most $2n-5$ circle events (because a new Voronoi vertex is detected)

Result

Theorem: The Voronoi diagram of a set of n point sites in the plane can be computed in $O(n \log n)$ time