Voroni Diagarms

The post office problem

Zhale tirgary

1391/1/28

HANDEL CIRCLE EVENT(γ)

1. Delete the leaf γ that represent the disappearing arc α from T. Update the tuple representing the breakpoints at the internal nodes.perform rebalancing operations on T if necessary. Delete all circle events involving α from Q; these can be found using the pointers from the predecessor and the successor of γ in T. (The circle event where α is the middle are is currently being handled, and has been deleted from Q.)

2. Add the center of the circle causing the event as a vertex record to the doubly-connected edge list D storing the Voronoi diagram under construction. Create two half-edge records corresponding to the new breakpoint of the beach line. Set the pointer between them appropriately. Attach the three new records to the half-edge records that end at the vertex.

3. Check the new triple of consecutive arcs that has the former left neighbor of α as its middle arc to see if the two breackpoints of the triple converage. If so, insert the corresponding circle event into Q. and set pointers between the new circle event in Q and the corresponding leaf of T. Do the same for the triple where the former right neighbor is the middle arc.

Lemma 7.9

The algorithm runs in O(nlogn) time and it uses O(n)storage.

proof.

The primitive operatians on the tree T and event queue Q, such that as inserting or deleting an element, take

O(logn) time each. The primitive operation on the doubly-connected adge list take constant time. To handle an event

We do a constant number of such primitive operation, so we spend O(logn) time to prcess an event. Obviously, there are n site events.Az for the number of circle event , we obseve that in o(n).

Degenerate Cases.

1.When two or more events lie on a common horizontal line. These event can be handled

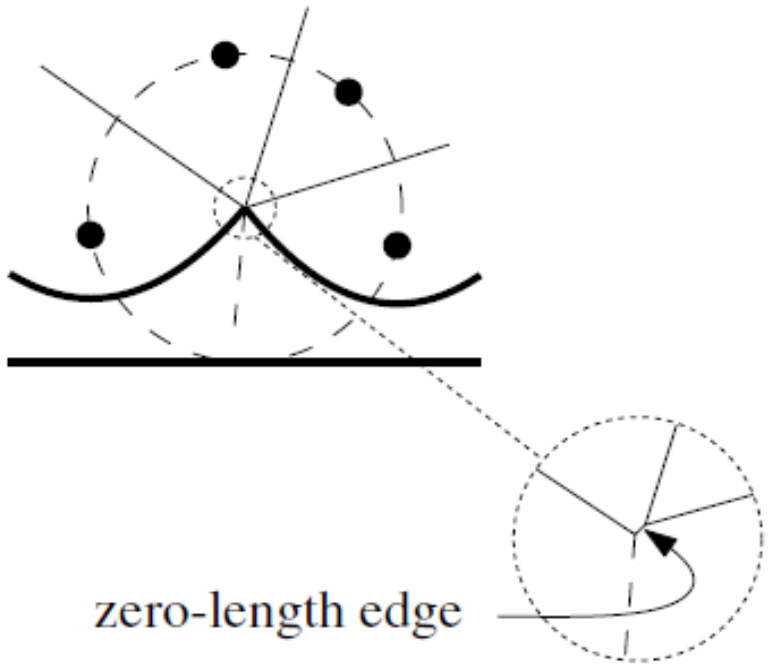
In any order when their x-coordinate are distinct.if this happen s right at the start of the

Algorithm then special code is needed.

2.there are event points that coincide.circle event envolving more than 3 site

Instead of producing a vertex whit degree four, it will just produce two vertices whit degree three

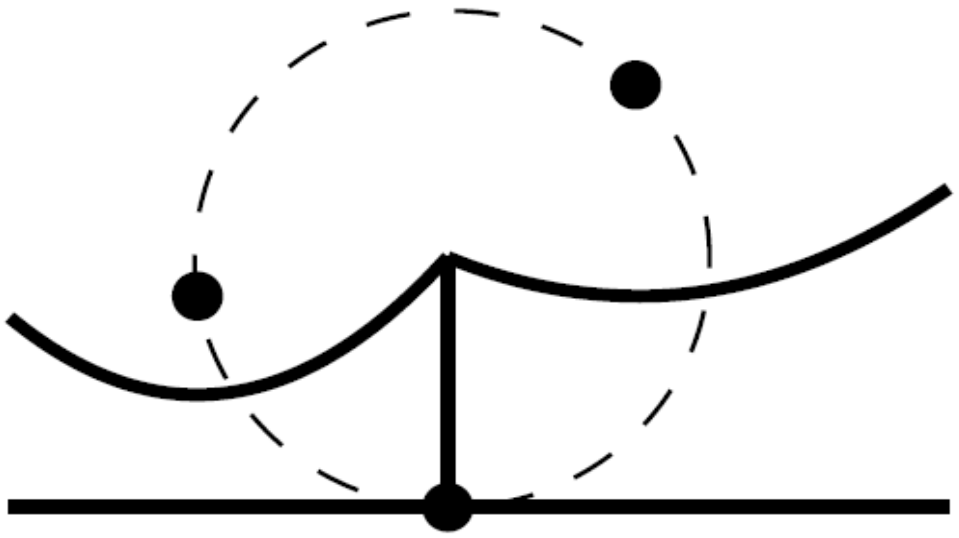
Whit a zero length adge between them.



3.When a site p_i that we process happens to be located exactly below the breakpoints between

Two arcs on the beach line,

Algorithm split either of these two arcs and inserts the arc for p_i in the between the two pieces, one of which has zero length.



4.Another degeneracy occurs when three consecutive arcs on the beach line are defined by

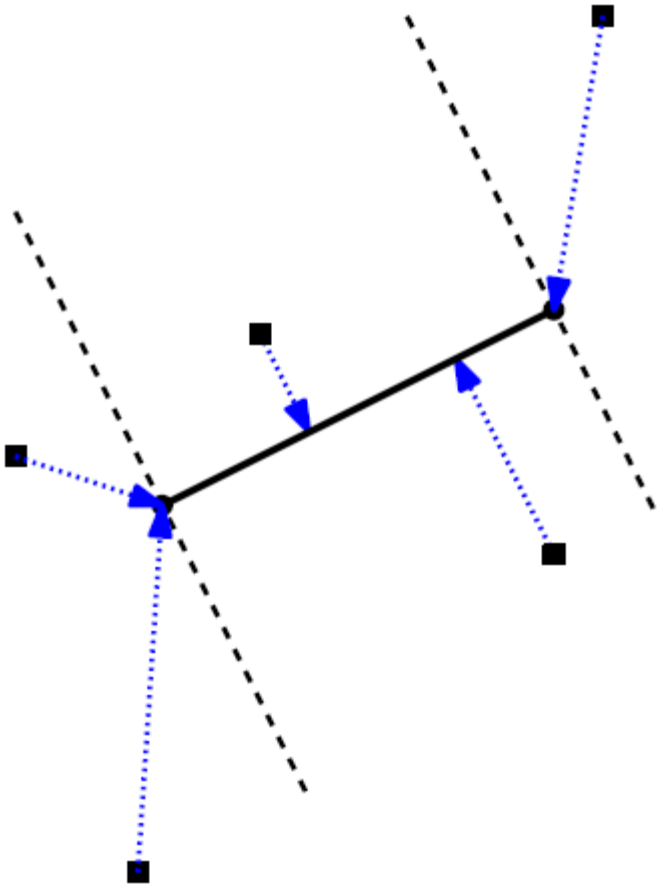
Three collinear sites. Then these sites don’t define

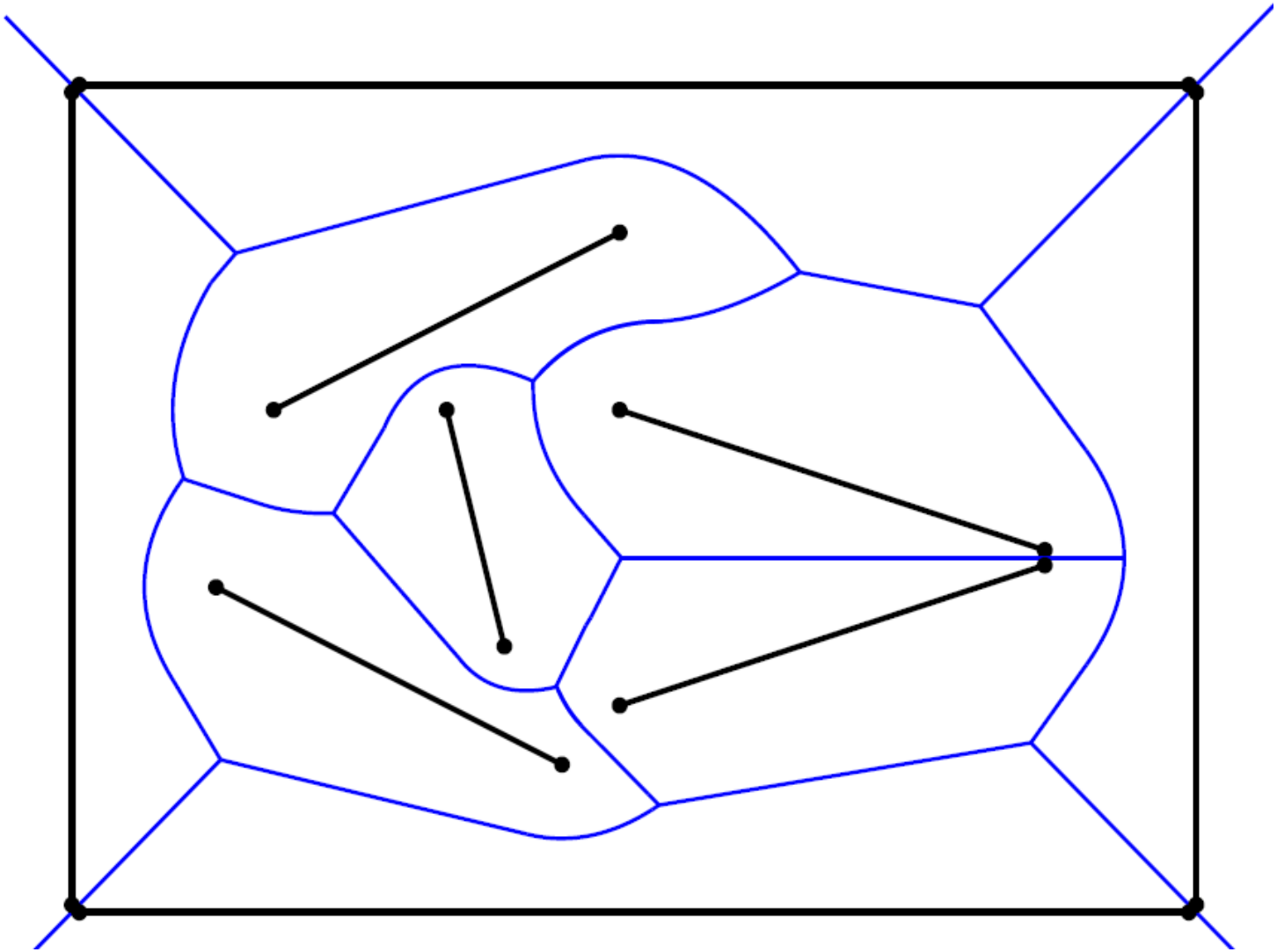
A circle, nor a circle event.

7.3 voronoi diagrams of line segment

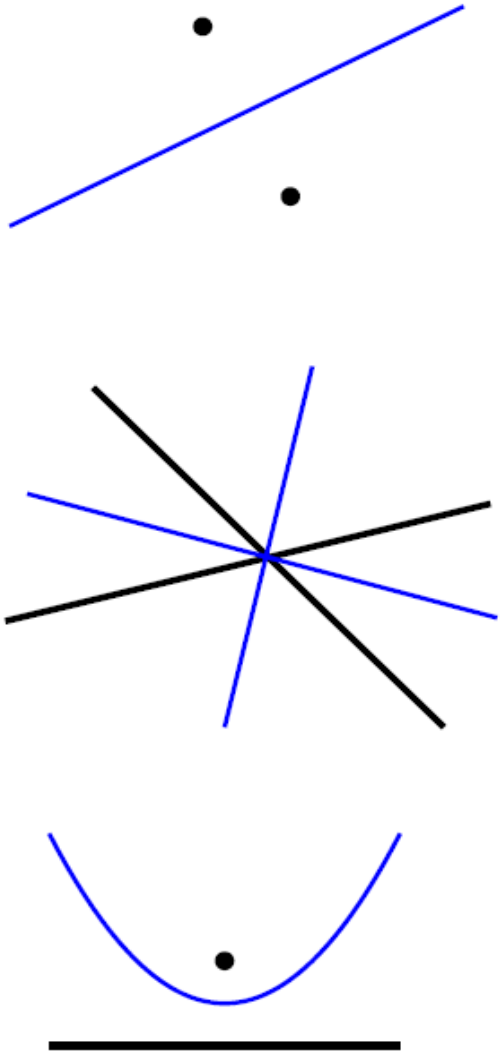
For a Voronoi diagram of other objects than point sites, we must decide to which point on each site we measure the distance

This will be the closest point on the site

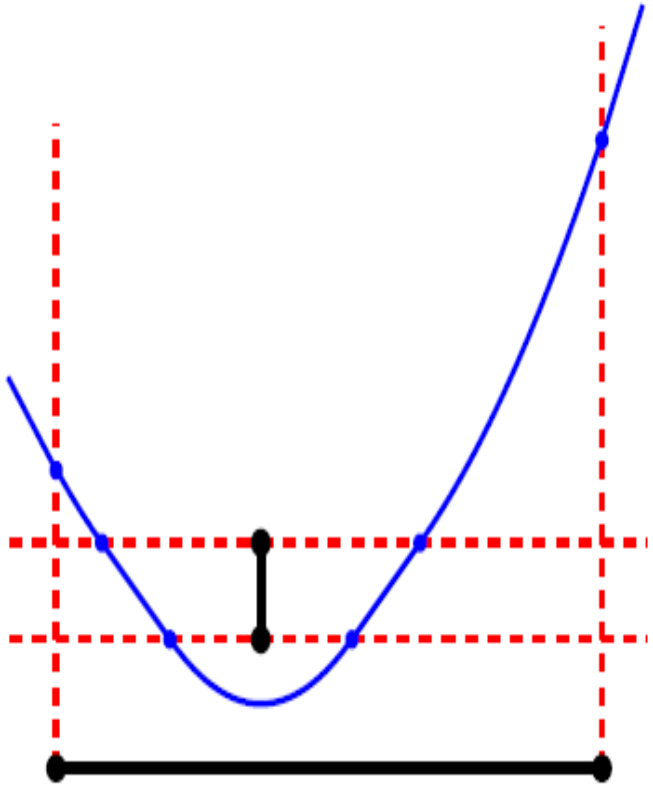




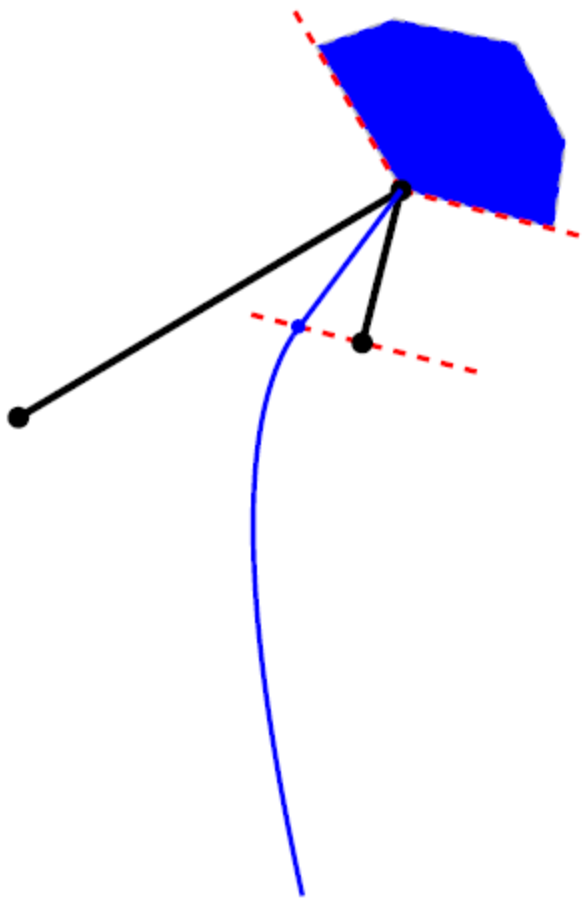
- The points of equal distance to two points lie on a line
- The points of equal distance to two lines lie on a line (two lines)
- The points of equal distance to a point and a line lie on a parabola



Two line segment sites have a bisector with up to 7 arcs



If two line segment sites share an endpoint, their bisector can have an area too



Definition

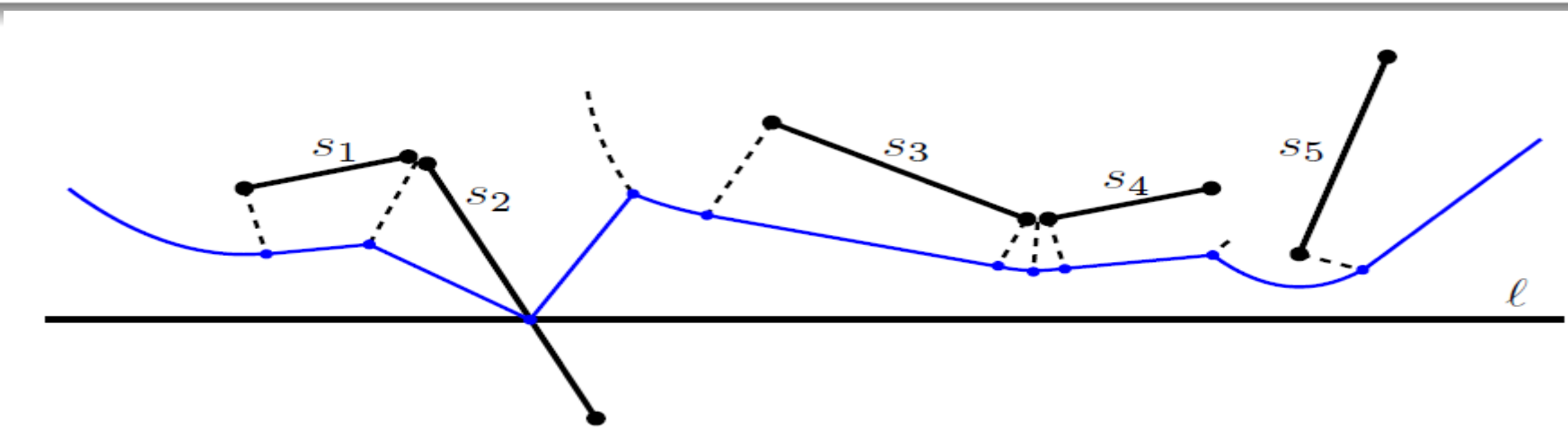
Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n disjoint line segments. We call the segments of the S sites as before, and use the terms *site endpoint* and *site interior*.

Recall

A beach line is picewise parabolic x-monotone curve, such that the distance to the closest site above the sweep line is equal to the distance to the sweep line .

What dose beach line look like?

- Note that a line segment site may be partially above and partially below the sweep line ,we consider those part of the site that are above the sweep line.
- The beach line consist of those points such that the distance to the closest portion of site above l is equal to distance to l.
- The beach line consist of parabolic arcs and straight line segments.



The algorithm uses 5 types of breakpoint:

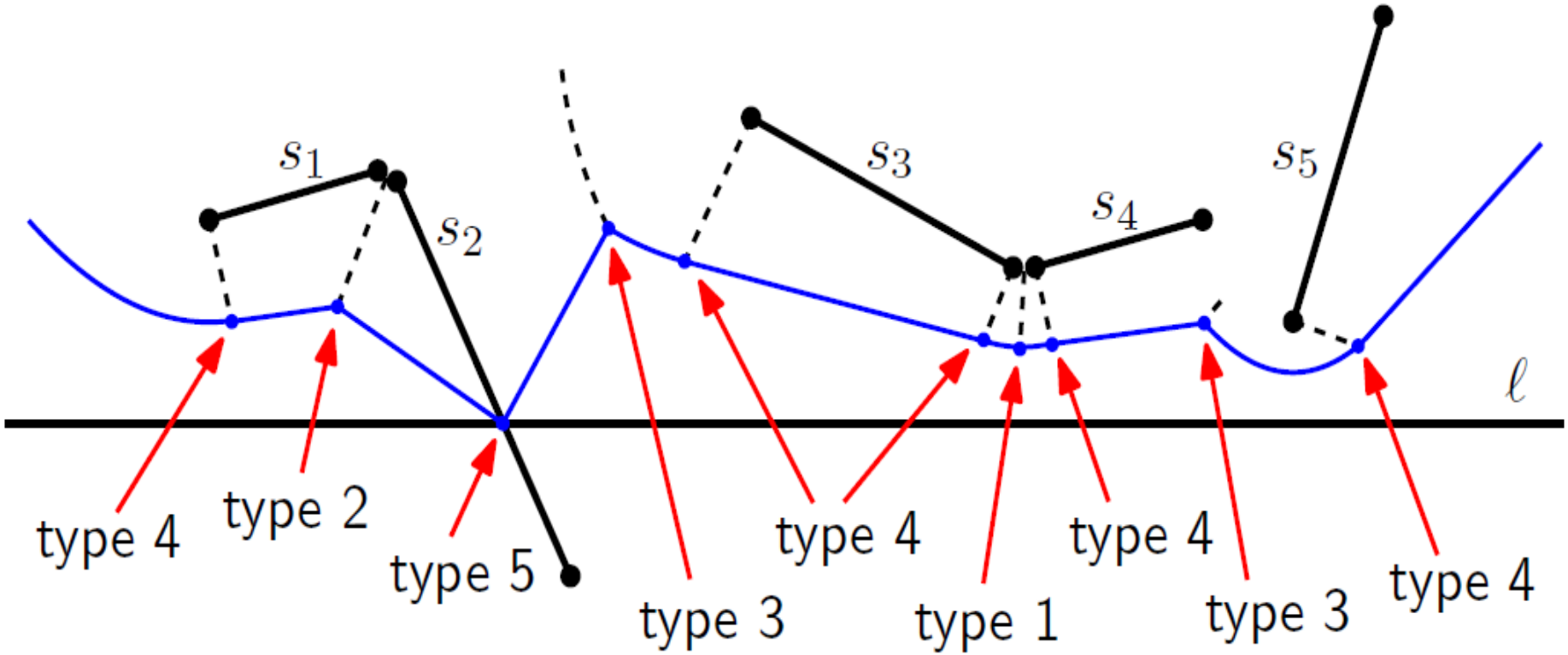
1. If a point p is closest to **two site endpoints** while being equidistant from them and ℓ, then p is a breakpoint that traces a line segment (as in the point site case)
2. If a point p is closest to **two site interiors** while being equidistant from them and ℓ, then p is a breakpoint that traces a line segment
3. If a point p is closest to **a site endpoint and a site interior** of different sites while being equidistant from them and ℓ, then p is a breakpoint that traces a parabolic arc

The algorithm uses 5 types of breakpoint (continued):

4. If a point p is closest to a site endpoint, the shortest distance is realized by a segment that is perpendicular to the line segment site, and p has the same distance from ℓ, then p is a breakpoint that traces a line segment

5. If a site interior intersects the sweep line, then the intersection is a breakpoint that traces a line segment (the site interior)

These two types of breakpoint do not trace Voronoi diagram edges but they do trace breaks in the beach line


s_1
s_2
s_3
s_4
s_5
ℓ
type 4
type 2
type 5
type 3
type 4
type 1
type 4
type 3
type 4

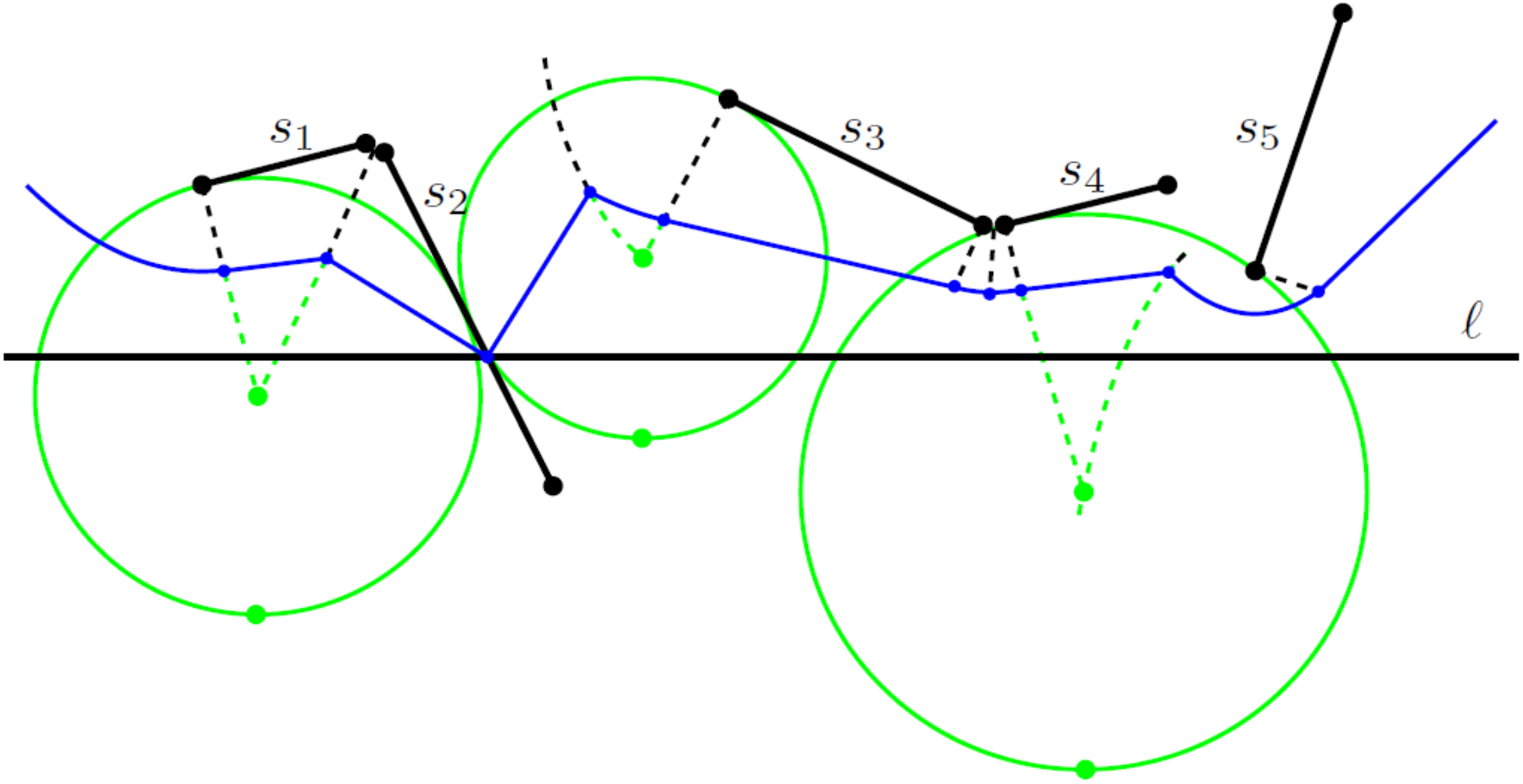
Types

There are site events and circle events

Site Event

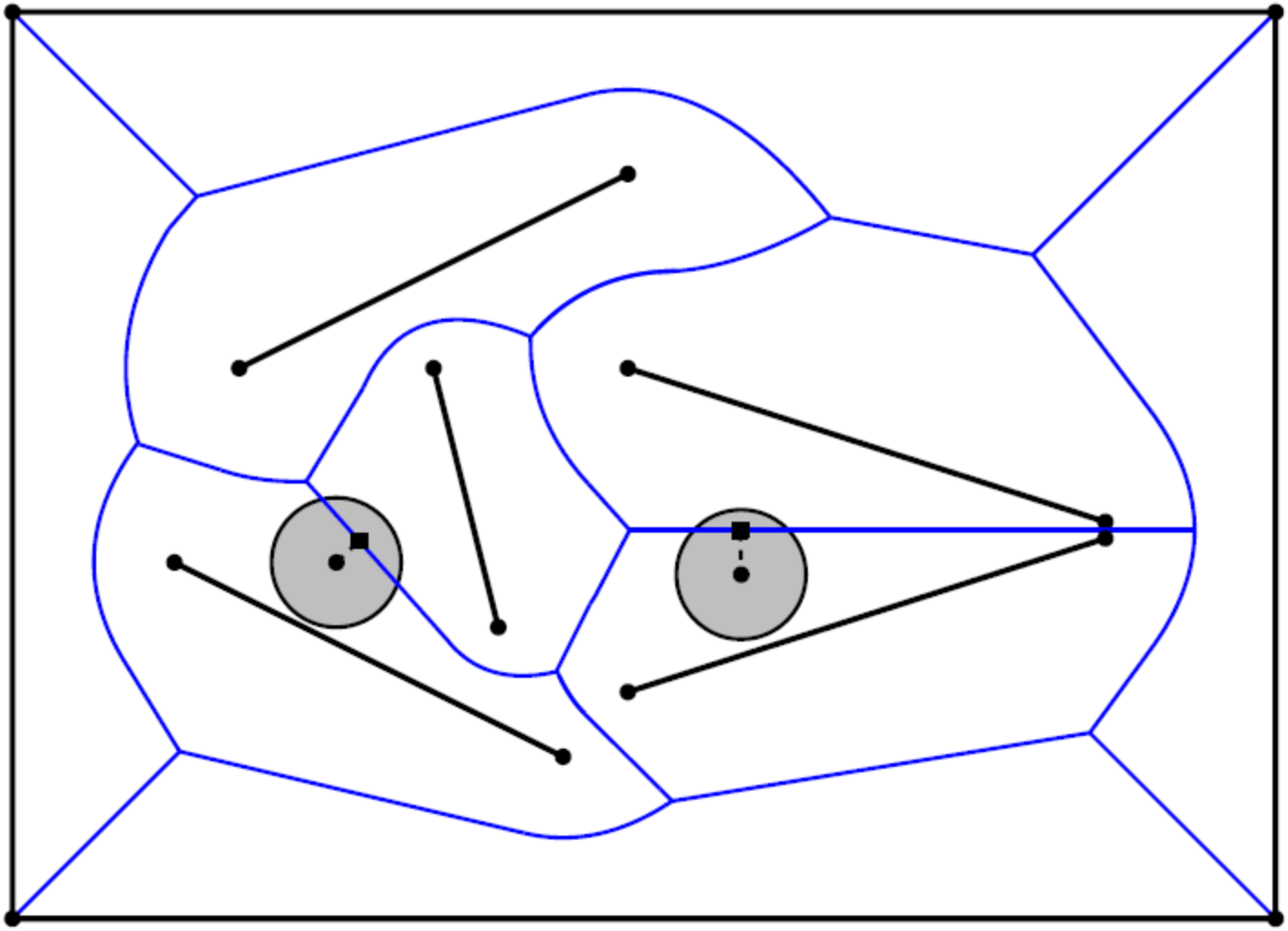
- It accurs when the sweep line reachs a site endpoint.
- Site events at upper endpoint handled differently from site events at lower endpoints :
  - At an upper endpoint, an arc of beach line is split into two ,and in between , four new arcs appear.
  - At a lower end point , the breakpoints that is intersection of the site enterior with sweep line is replaced by two breakpoints of the fourth type

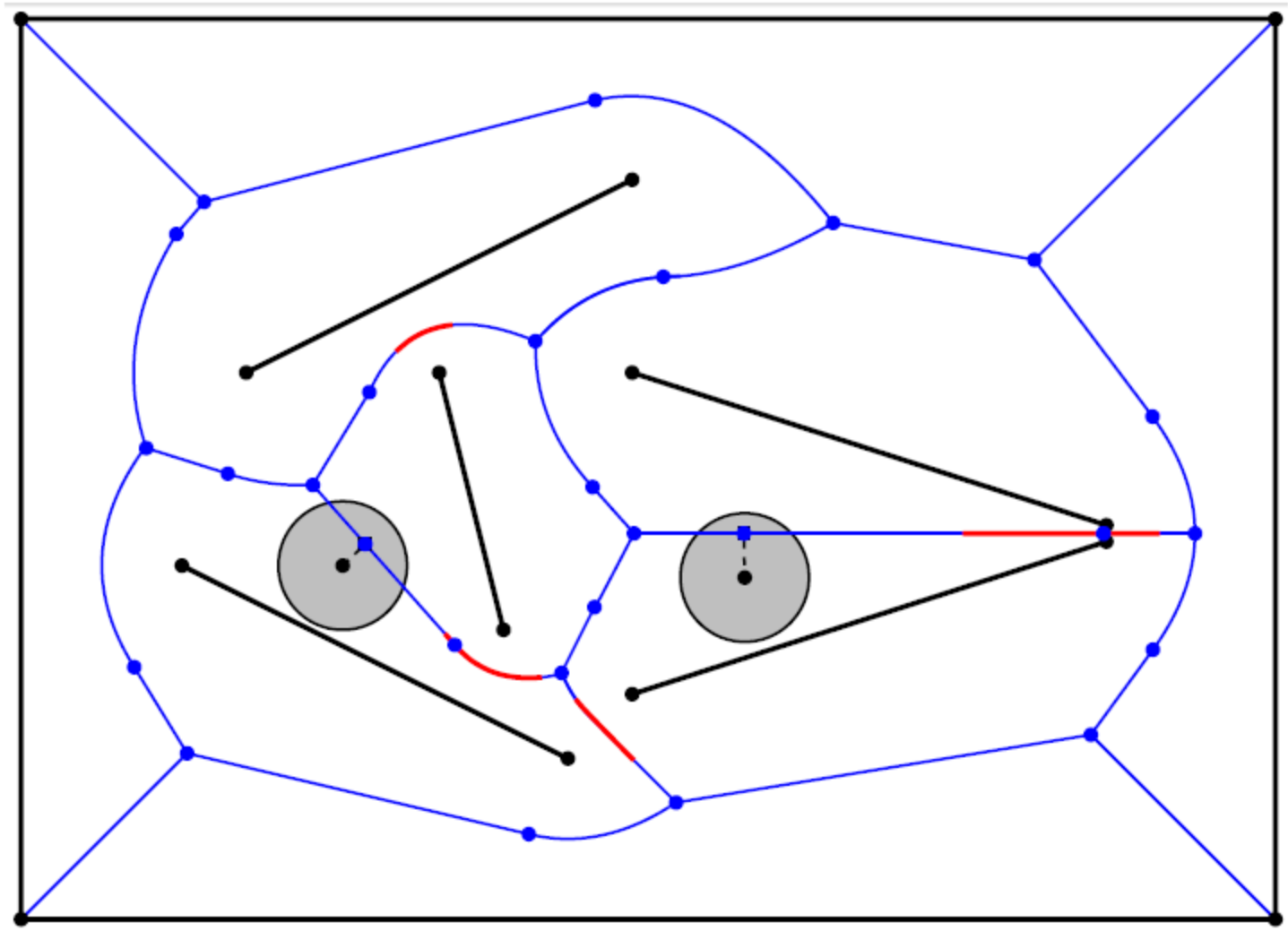
- There are several types of circle event
- They all correspond to the disappearance of an arc on the beach line
- The center of these empty circles arc where two consecutive breakpoints will meet
- They accur when the sweep line reachs the bottom of an empty circle that is defined by two or three site above the sweep line
- The types of circle events essentially correspond to the types of breakpoints that meet

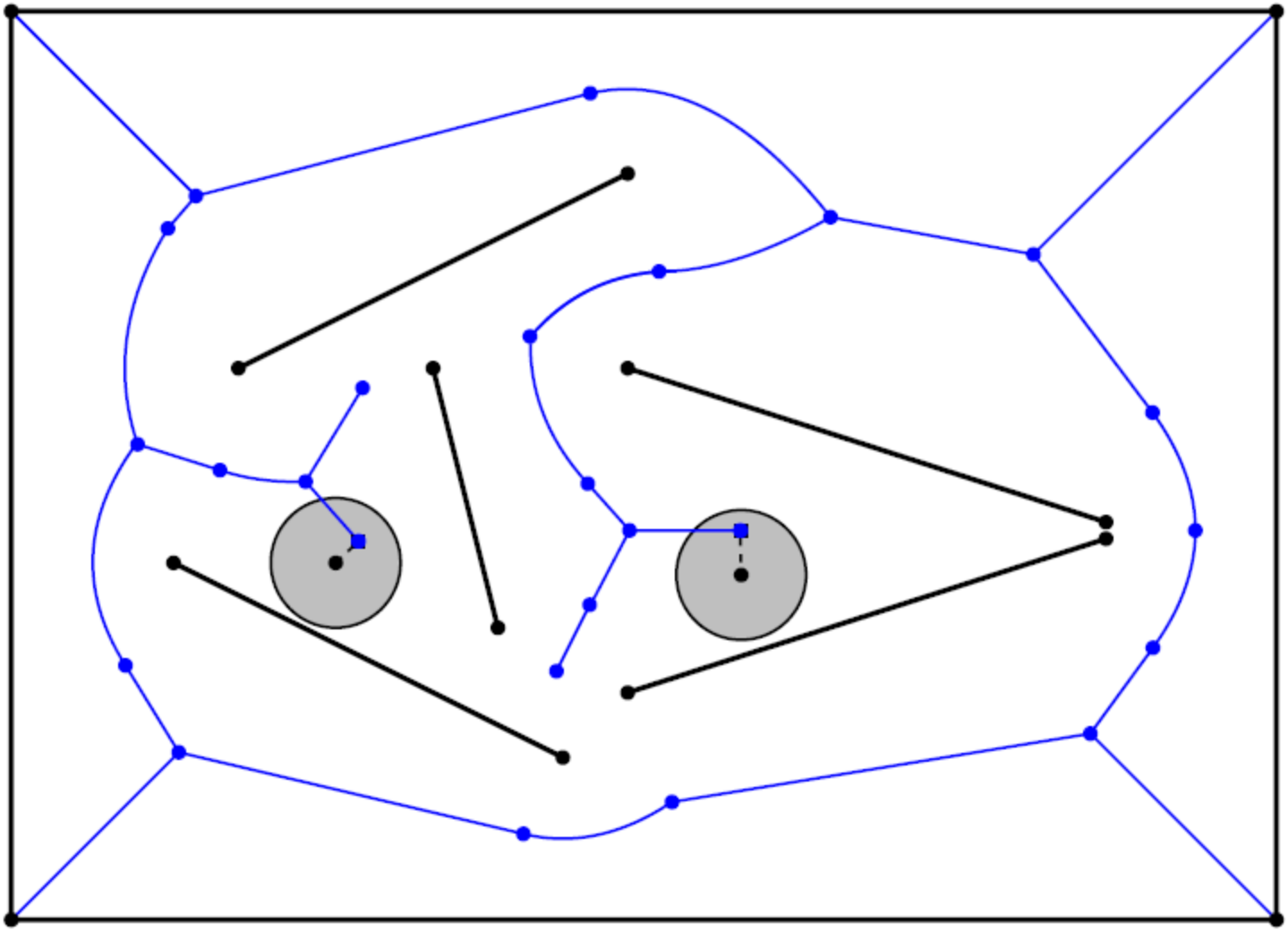

s_1
s_2
s_3
s_4
s_5
ℓ

Theorem 7.11

The Voronoi diagram of a set of n disjoint line segment sites can be computed in $O(n \lg n)$ time using $O(n)$ storage







Algorithm RETRACTION$(S, q_{\text{start}}, q_{\text{end}}, r)$

Input. A set $S := \{s_1, \ldots, s_n\}$ of disjoint line segments in the plane, and two discs D_{start} and D_{end} centered at q_{start} and q_{end} with radius r. The two disc positions do not intersect any line segment of S.

Output. A path that connects q_{start} to q_{end} such that no disc of radius r with its center on the path intersects any line segment of S. If no such path exists, this is reported.

1. Compute the Voronoi diagram $\text{Vor}(S)$ of S inside a sufficiently large bounding box.
2. Locate the cells of $\text{Vor}(P)$ that contain q_{start} and q_{end}.
3. Determine the point p_{start} on $\text{Vor}(S)$ by moving q_{start} away from the nearest line segment in S. Similarly, determine the point p_{end} on $\text{Vor}(S)$ by moving q_{end} away from the nearest line segment in S. Add p_{start} and p_{end} as vertices to $\text{Vor}(S)$, splitting the arcs on which they lie into two.
4. Let $\mathcal{G}$ be the graph corresponding to the vertices and edges of the Voronoi diagram. Remove all edges from $\mathcal{G}$ for which the smallest distance to the nearest sites is smaller than or equal to r.
5. Determine with depth-first search whether a path exists from p_{start} to p_{end} in $\mathcal{G}$. If so, report the line segment from q_{start} to p_{start}, the path in $\mathcal{G}$ from p_{start} to p_{end}, and the line segment from p_{end} to q_{end} as the path. Otherwise, report that no path exists.

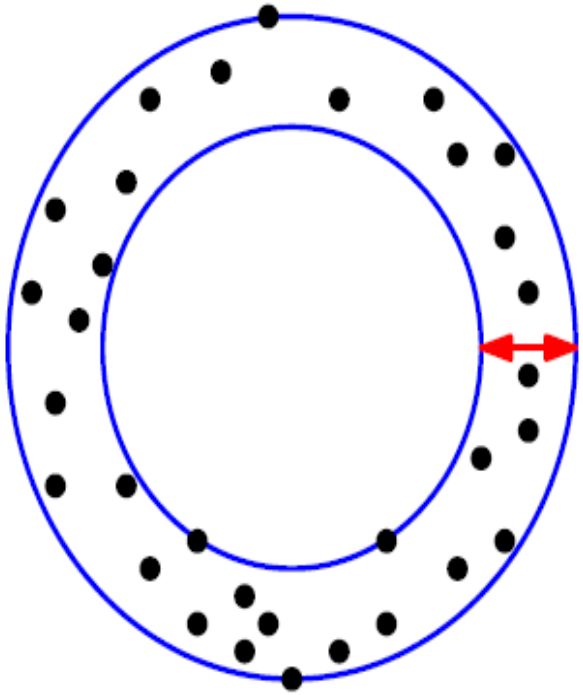
Theorem 7.12

Given n disjoint line segment obstacles and a disc-shaped robot, the existence of a collision-free path between two positions of the robot can be determined in $O(n \lg n)$ time using $O(n)$ storage

The roundness of a set of points is the width of the smallest annulus that contains the points

An annulus is the region between two co-centric circles

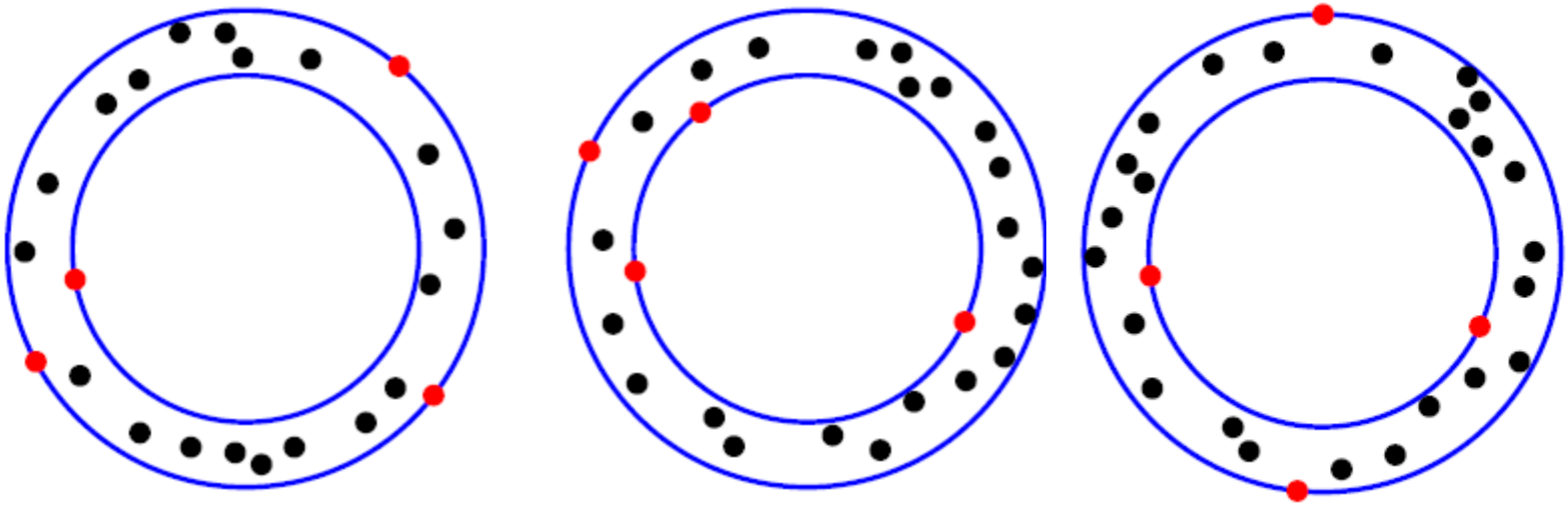
Its width is the difference in radius



The smallest-width annulus must have at least one point on C_{outer}, or else we can decrease its size and decrease the width

The smallest-width annulus must have at least one point on C_{inner}, or else we can increase its size and decrease the width

- C_{outer} contains at least three points of P, and C_{inner} contains at least one point of P
- C_{outer} contains at least one point of P, and C_{inner} contains at least three points of P
- C_{outer} and C_{inner} both contain two points of P



Finding the smallest-width annulus is equivalent to finding its center point.

On the center point –let's call it q –is fixed.
The annulus is determined by the point of p that
Are closest to and farthest from q.

If we have the voronoi diagram of p, then the closest
Point is the one in whose cell q lies.

It turns out that a similar structure exists
For the farthest point, namely the farthest-
Point diagram.

The farthest –point voronoi cell of a point
p_i is the intersection of n-1 half-planes
Just as for a standard voronoi cell, but
We take the “other sides” of bisectors

Observation 7.13

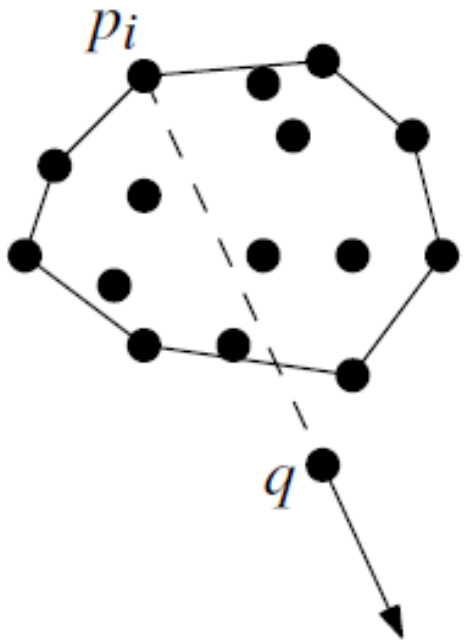
Given a set P of points in the plane, a point of P has a cell In the farthest-point voronoi diagram if and only if it is a Vertex of the convex hull of P.

More properties of the farthest-point voronoi diagram

Suppose that a point $p_i \in P$ lies on the convex hull

And q be some point in the plane for which p_i is the

Farthest point .



Let $l(p_i, q)$, be the line through p_i and q. then all Points on the half-line starting at q, contained in $l(p_i, q)$, and not containing p_i, must also be in the Farthest-point voronoi cell of p_i.

This implies that all cells are unbounded.

The simplest algorithm to construct the farthest-point Voronoi diagram is randomized incremental construction on the convex hull vertices

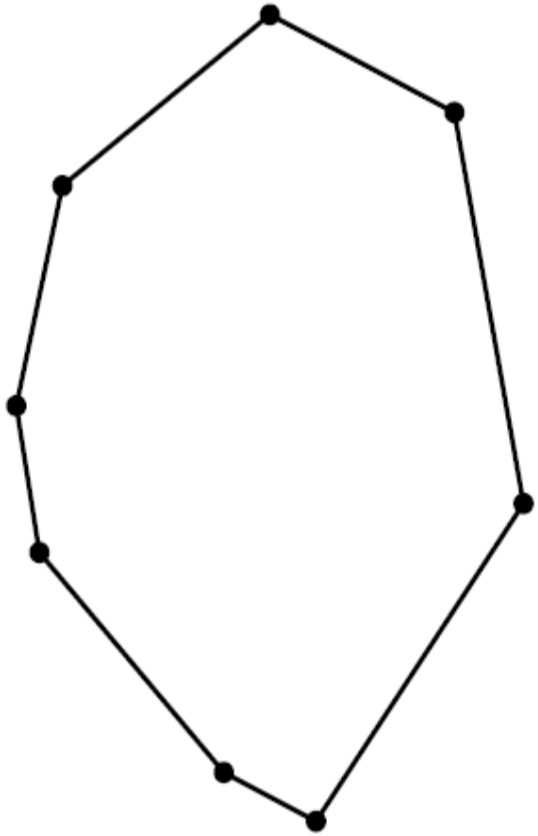
Let $p_1, \ldots, p_m$ be the points in *random order*

From the convex hull, we also know the *convex hull order*

Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

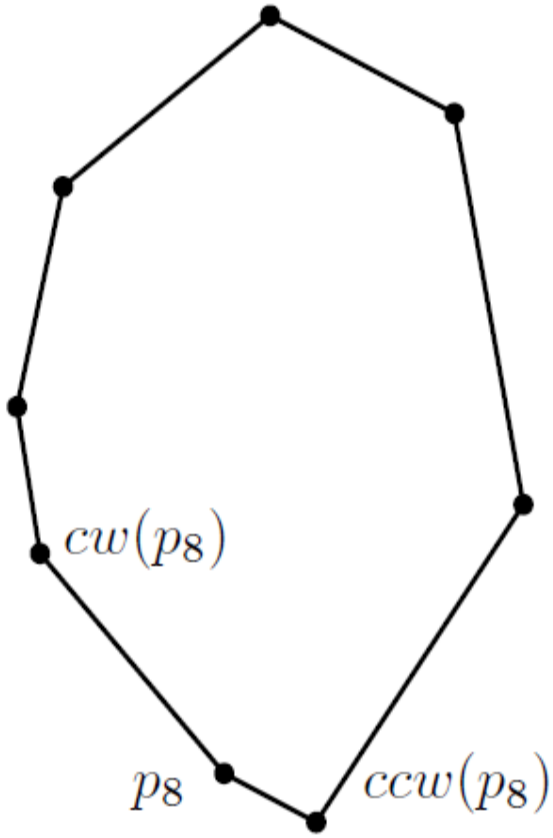
Remove p_i from the convex hull;
remember its 2 neighbors $cw(p_i)$
and $ccw(p_i)$ (at removal!)



Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

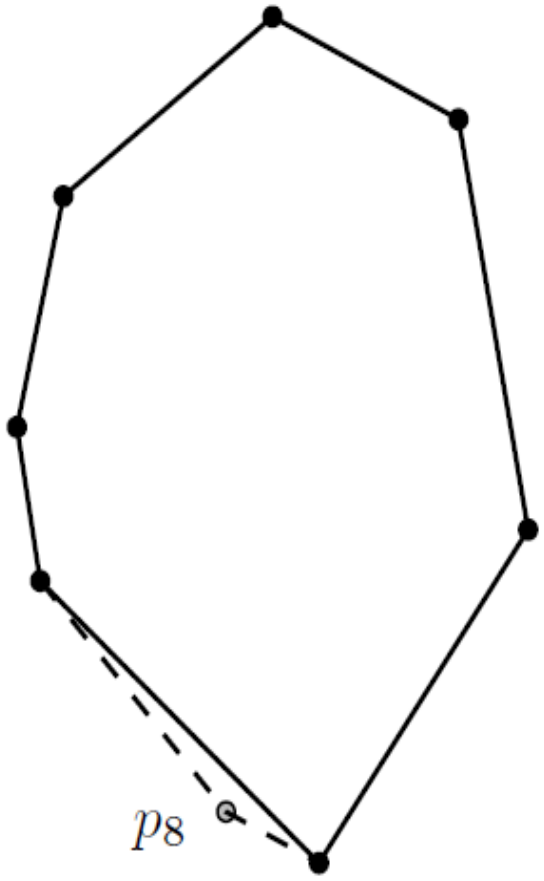
Remove p_i from the convex hull;
remember its 2 neighbors $cw(p_i)$
and $ccw(p_i)$ (at removal!)



Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

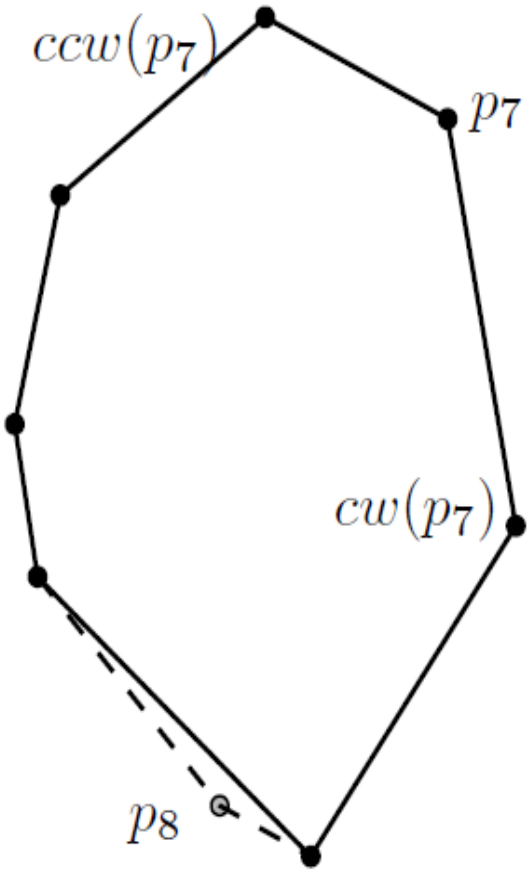
Remove p_i from the convex hull;
remember its 2 neighbors $cw(p_i)$
and $ccw(p_i)$ (at removal!)



Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

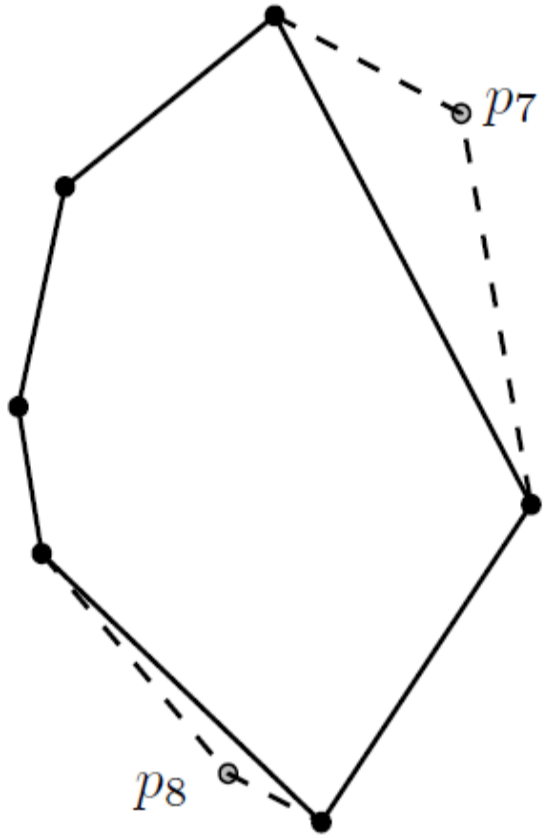
Remove p_i from the convex hull;
remember its 2 neighbors $cw(p_i)$
and $ccw(p_i)$ (at removal!)



Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

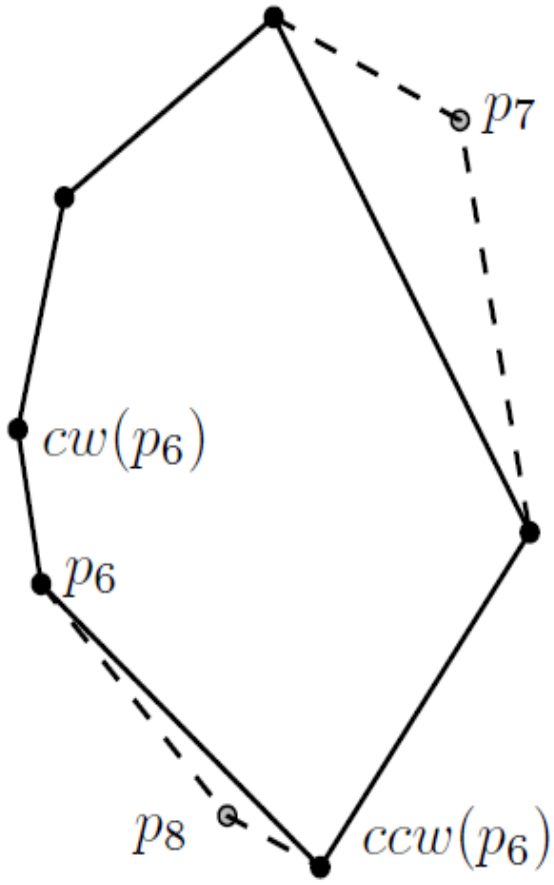
Remove p_i from the convex hull;
remember its 2 neighbors $cw(p_i)$
and $ccw(p_i)$ (at removal!)



Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

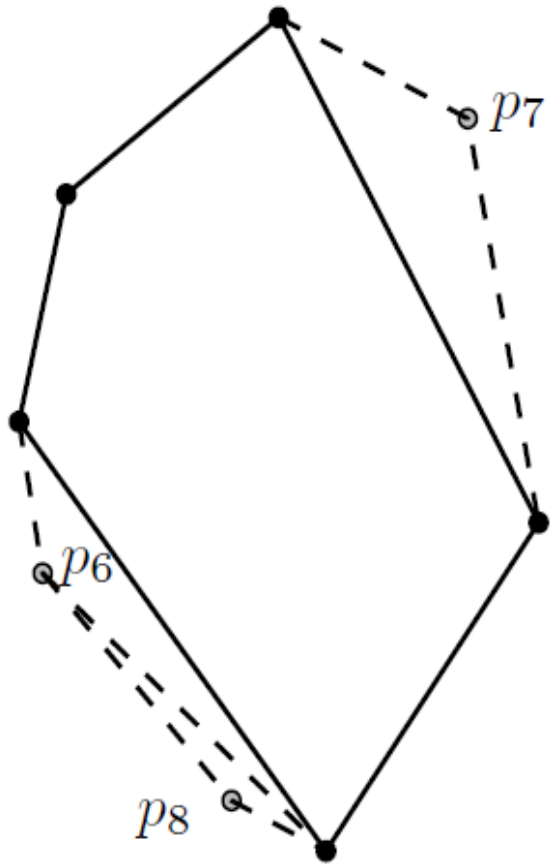
Remove p_i from the convex hull; remember its 2 neighbors $cw(p_i)$ and $ccw(p_i)$ (at removal!)



Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

Remove p_i from the convex hull;
remember its 2 neighbors $cw(p_i)$
and $ccw(p_i)$ (at removal!)



Phase 1: Remove and Remember

For $i \leftarrow m$ downto 4 do

Remove p_i from the convex hull; remember its 2 neighbors $cw(p_i)$ and $ccw(p_i)$ (at removal!)

$p_8, cw(p_8), ccw(p_8)$

$p_7, cw(p_7), ccw(p_7)$

$p_6, cw(p_6), ccw(p_6)$

$p_5, cw(p_5), ccw(p_5)$

$p_4, cw(p_4), ccw(p_4)$

p_3, p_2, p_1

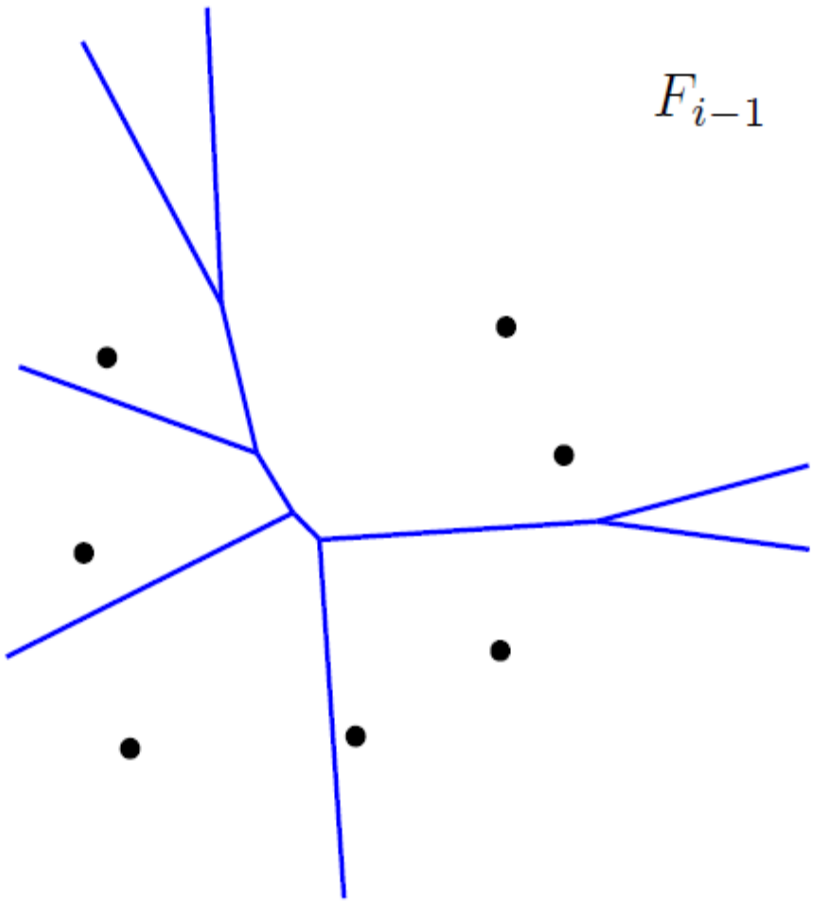
Phase 2: Put back and Construct

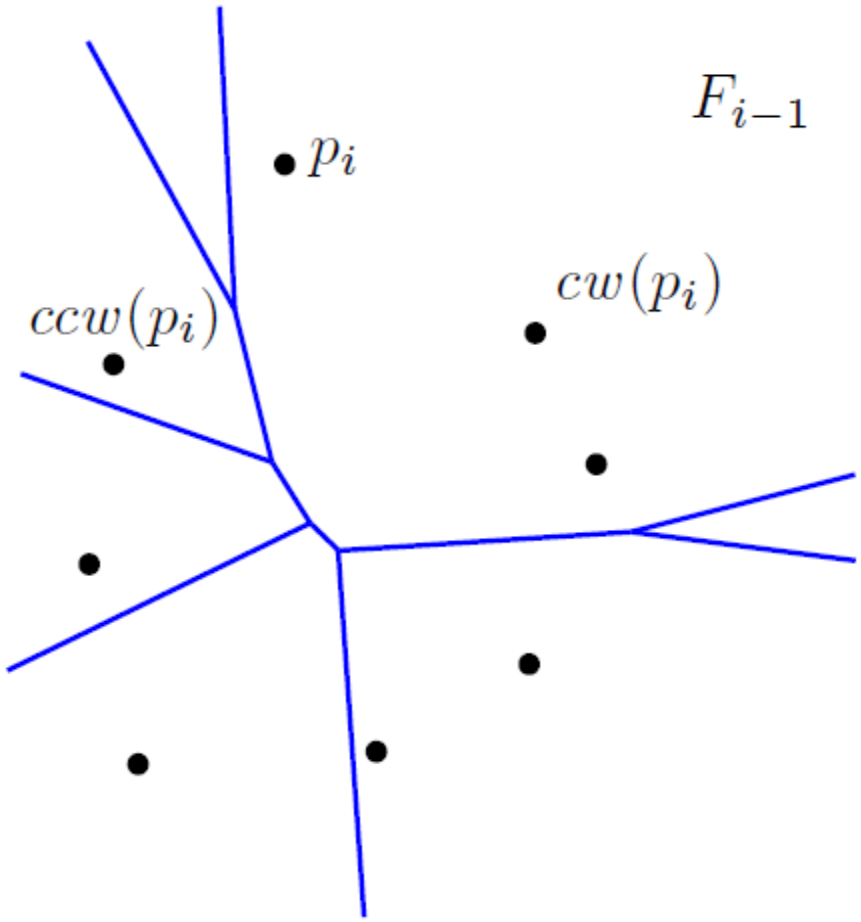
Construct the farthest-point Voronoi diagram F_3 of p_3, p_2, p_1

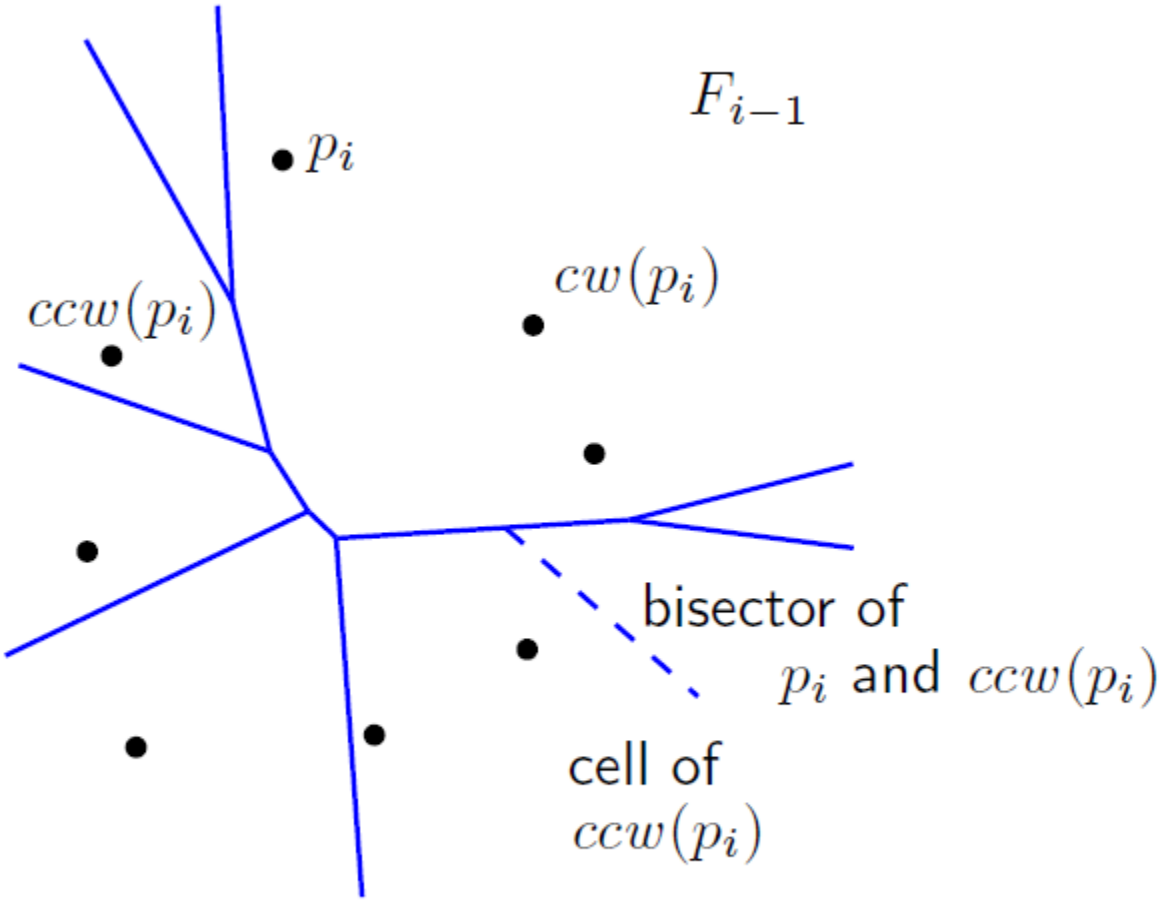
For $i \leftarrow 4$ to m do

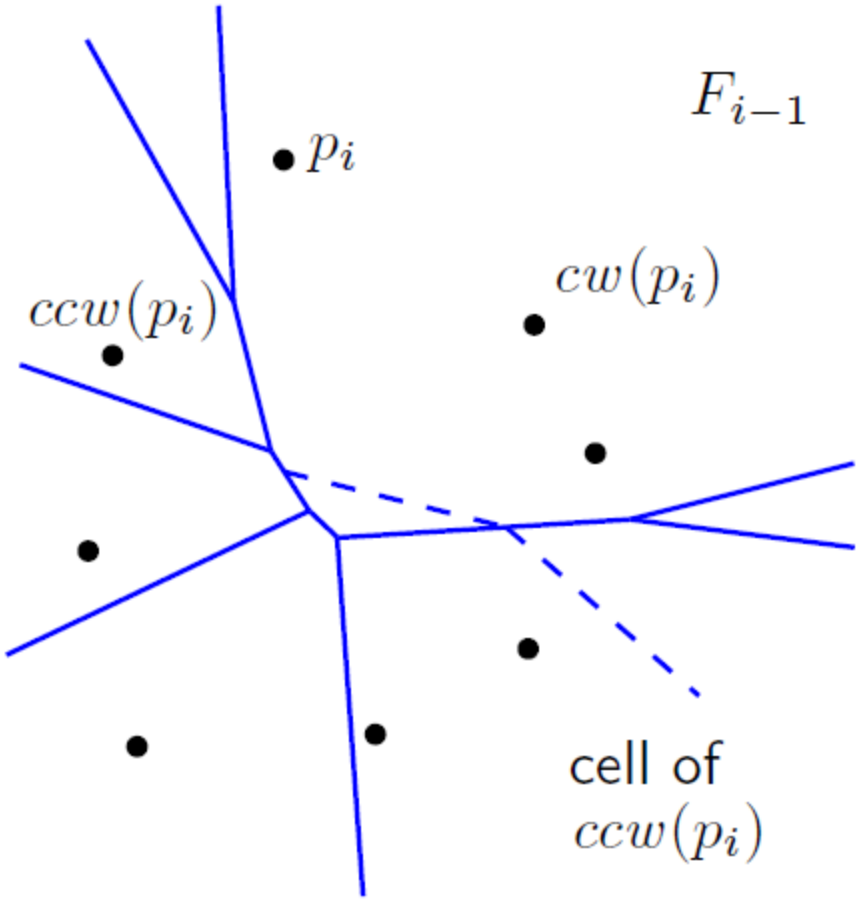
> Add p_i to the farthest-point Voronoi diagram F_{i-1} to make F_i

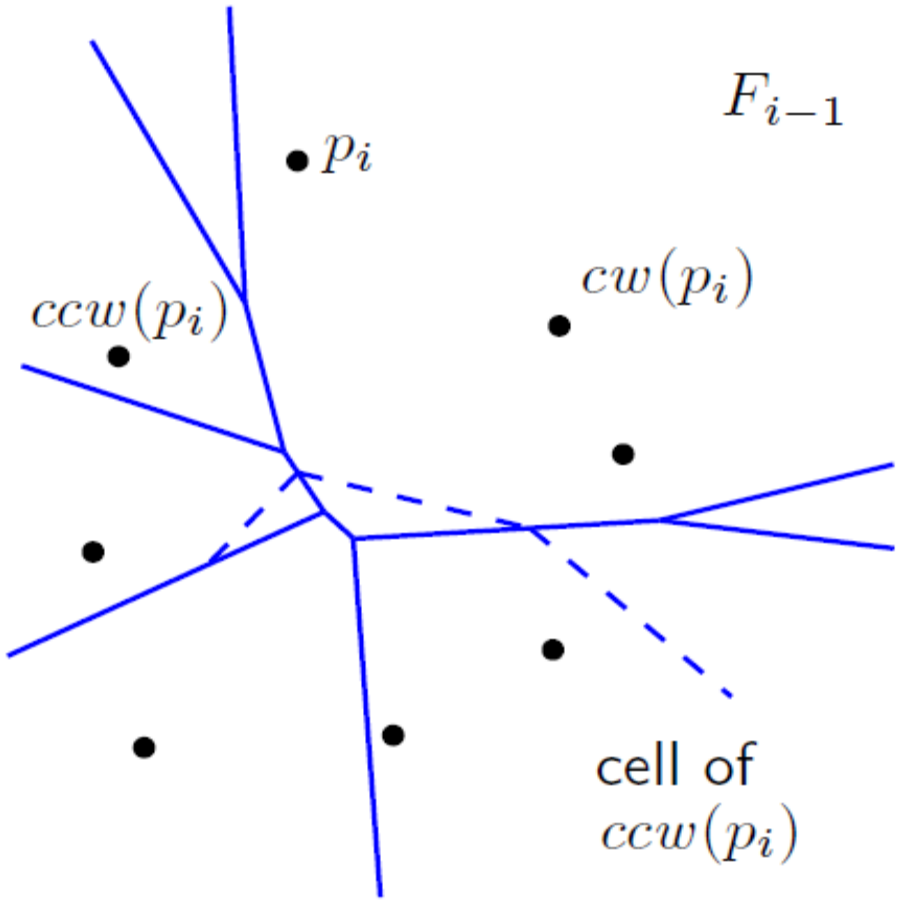
We simply determine the cell of p_i by traversing F_{i-1} and update F_{i-1}

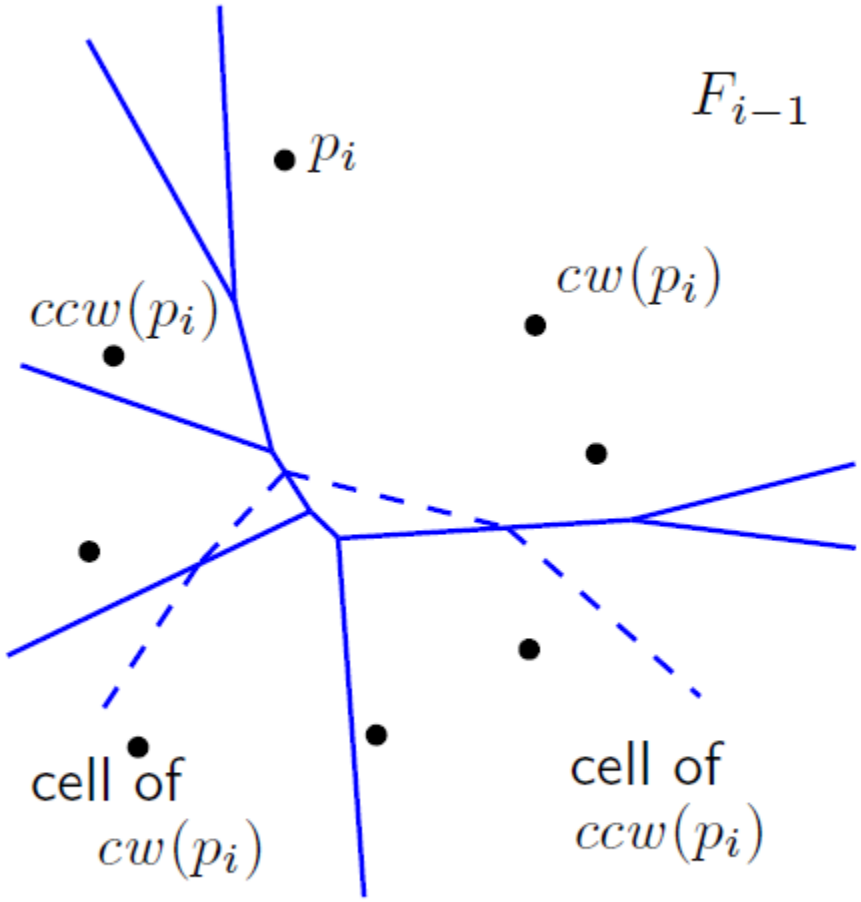

F_{i-1}

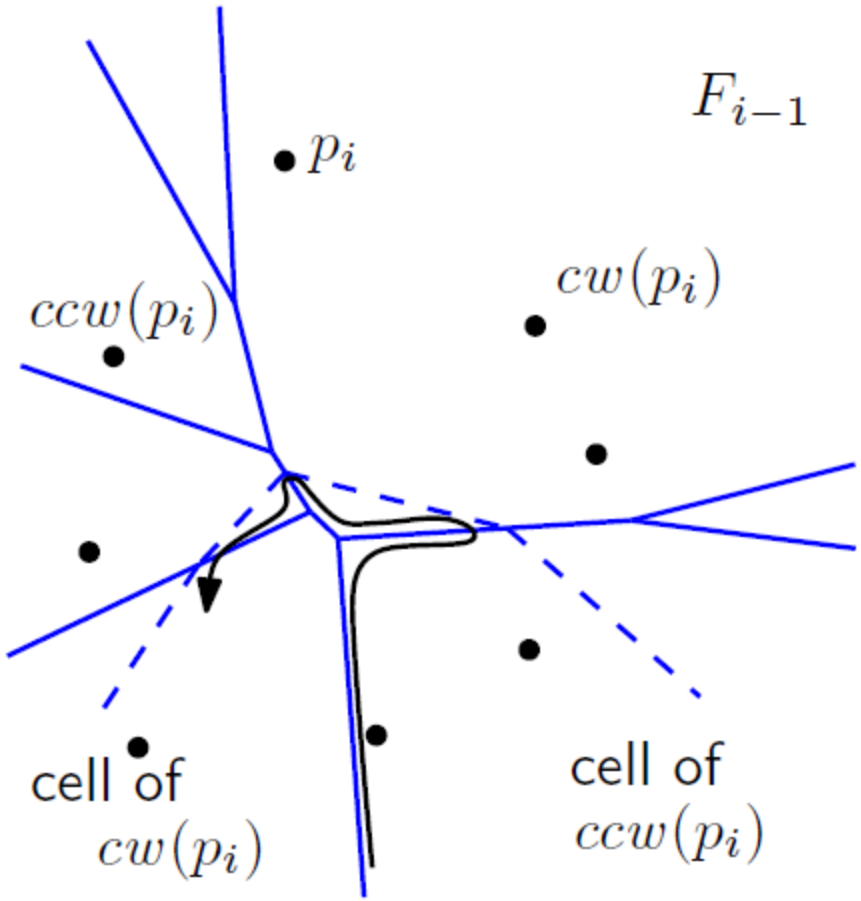

F_{i-1}
p_i
$cw(p_i)$
$ccw(p_i)$

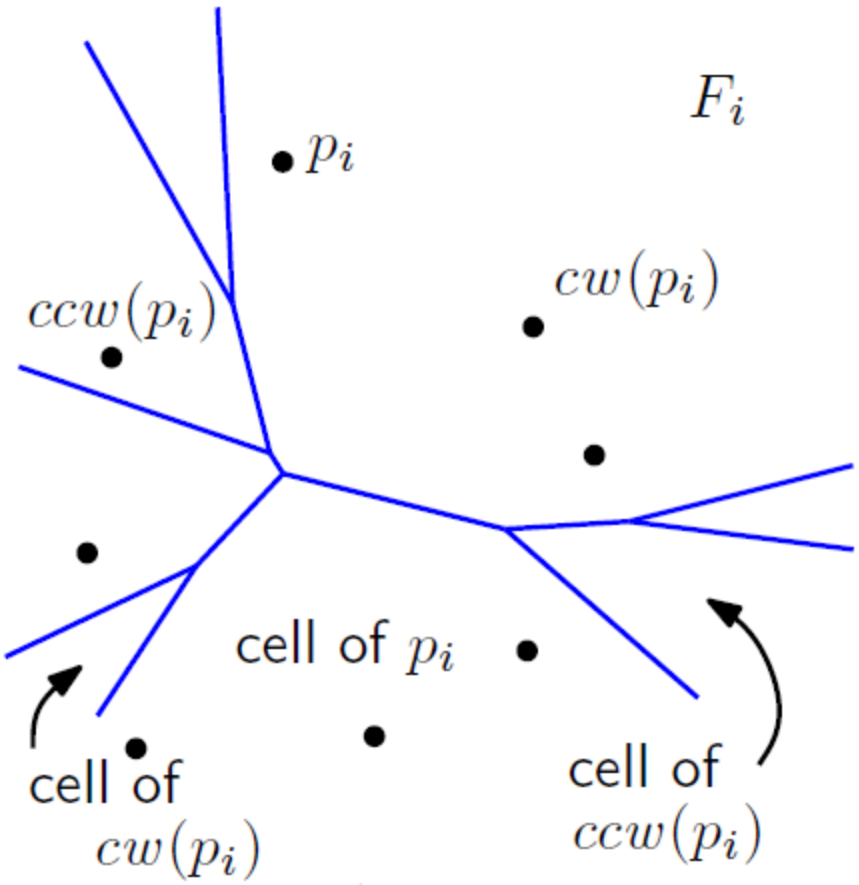

F_{i-1}
p_i
$cw(p_i)$
$ccw(p_i)$
bisector of
p_i and $ccw(p_i)$
cell of
$ccw(p_i)$


F_{i-1}
p_i
$cw(p_i)$
$ccw(p_i)$
cell of
$ccw(p_i)$

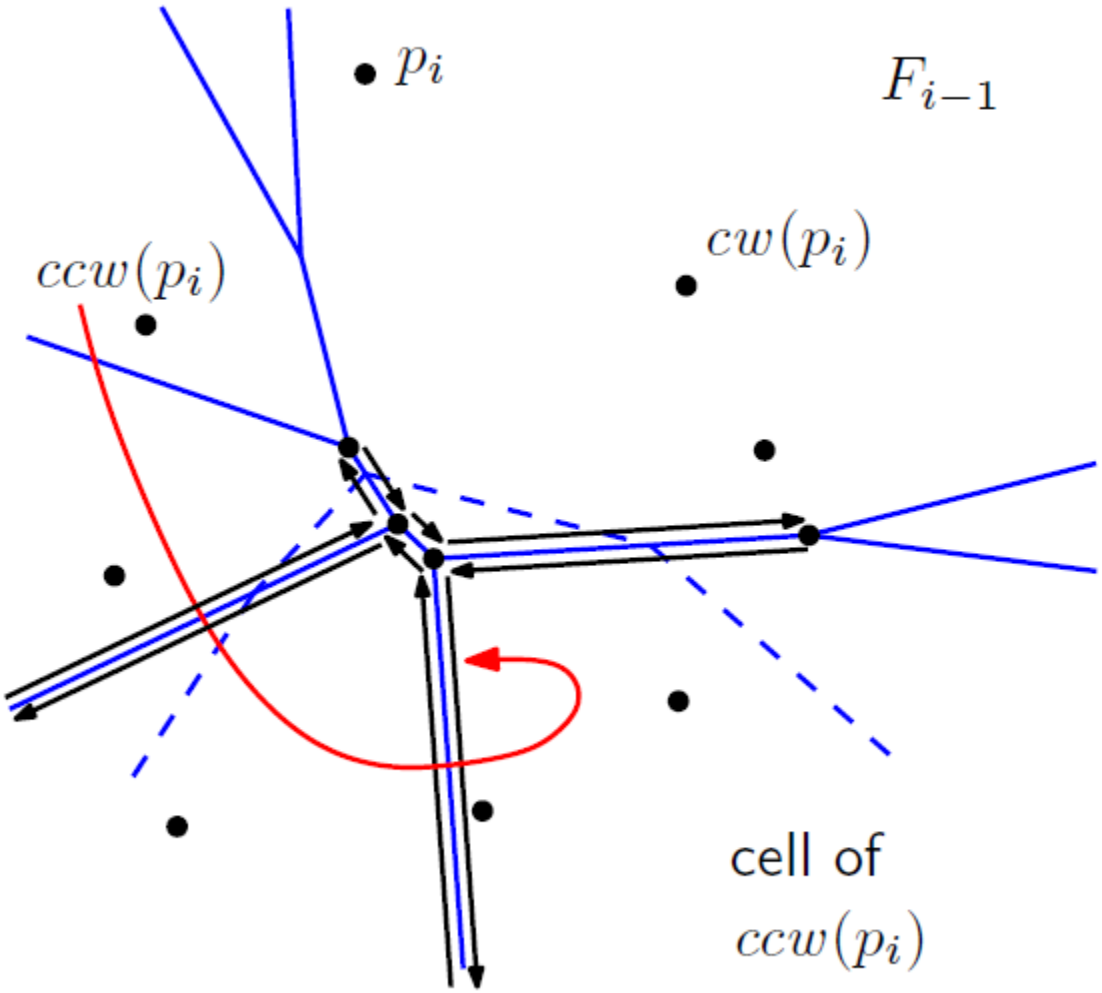

F_{i-1}
p_i
$cw(p_i)$
$ccw(p_i)$
cell of
$ccw(p_i)$


F_{i-1}
p_i
$cw(p_i)$
$ccw(p_i)$
cell of
$cw(p_i)$
cell of
$ccw(p_i)$


F_{i-1}
p_i
$ccw(p_i)$
$cw(p_i)$
cell of
$cw(p_i)$
cell of
$ccw(p_i)$


F_i
p_i
$cw(p_i)$
$ccw(p_i)$
cell of p_i
cell of $cw(p_i)$
cell of $ccw(p_i)$

For any point among $p_1, \ldots, p_{i-1}$, we maintain a pointer to the most counterclockwise bounding half-edge of its cell


p_i
F_{i-1}
$cw(p_i)$
$ccw(p_i)$
cell of
$ccw(p_i)$

Theorem 7.14

Given a set of n point in the plane, its farthest-point voronoi diagram can be computed in O(n logn) expected time using O(n) storage.

Proof.

it take O(n log n) time to compute the h points

On the convex hull. the farthest-point voroni diagram

Takes only O(h) expected time to construct after we

Have the point on the convex hull.

We applay backward analysis

We observe that if the cell p_i has k edges on its boundary

then the traversal performed to trace this cell visited k cells

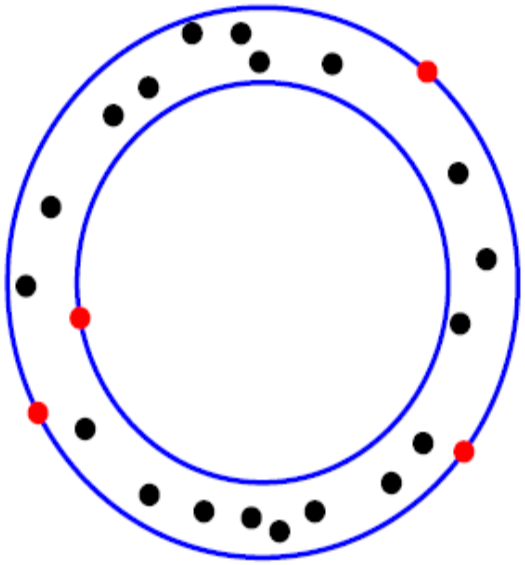
In the farthest-point voronoi diagram of $\{p_1, \ldots., p_{i-1}\}$, and

Visited at most 4k-6 boundary edges of these cells in total.

The expected size for cell p_i is less than four, hence, the expected time needed for each insert is O(1), and the algorithm runs in O(h) expected time.

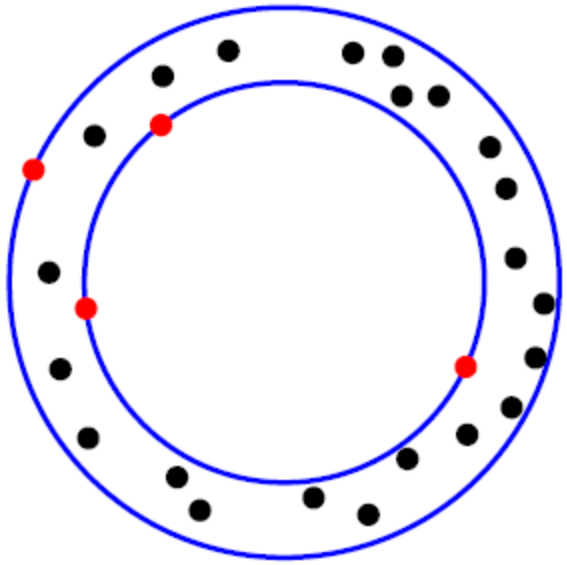
Consider **case 1**: C_{outer} contains (at least) three points of P and C_{inner} only one

Then the three points on C_{outer} define a "full" circle, and the center of C_{outer} is a farthest-point Voronoi diagram vertex!

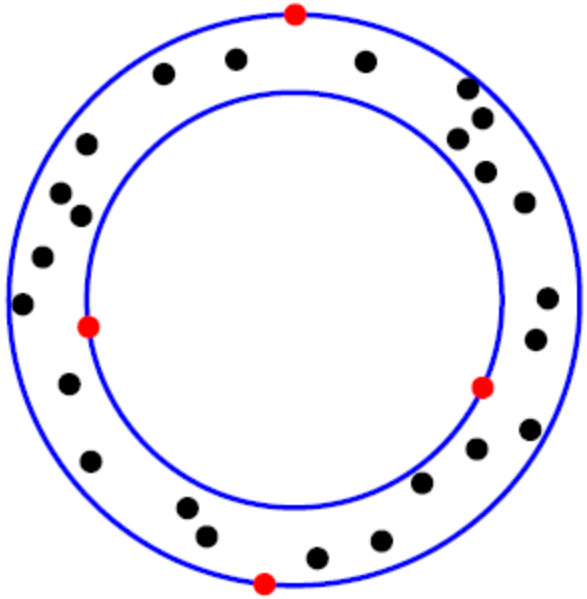


Consider **case 2**: C_{inner} contains (at least) three points of P and C_{outer} only one

Then the three points on C_{inner} define an empty circle, and the center of C_{inner} is a Voronoi diagram vertex!



Consider **case 3**: C_{outer} and C_{inner} each contain two points of P



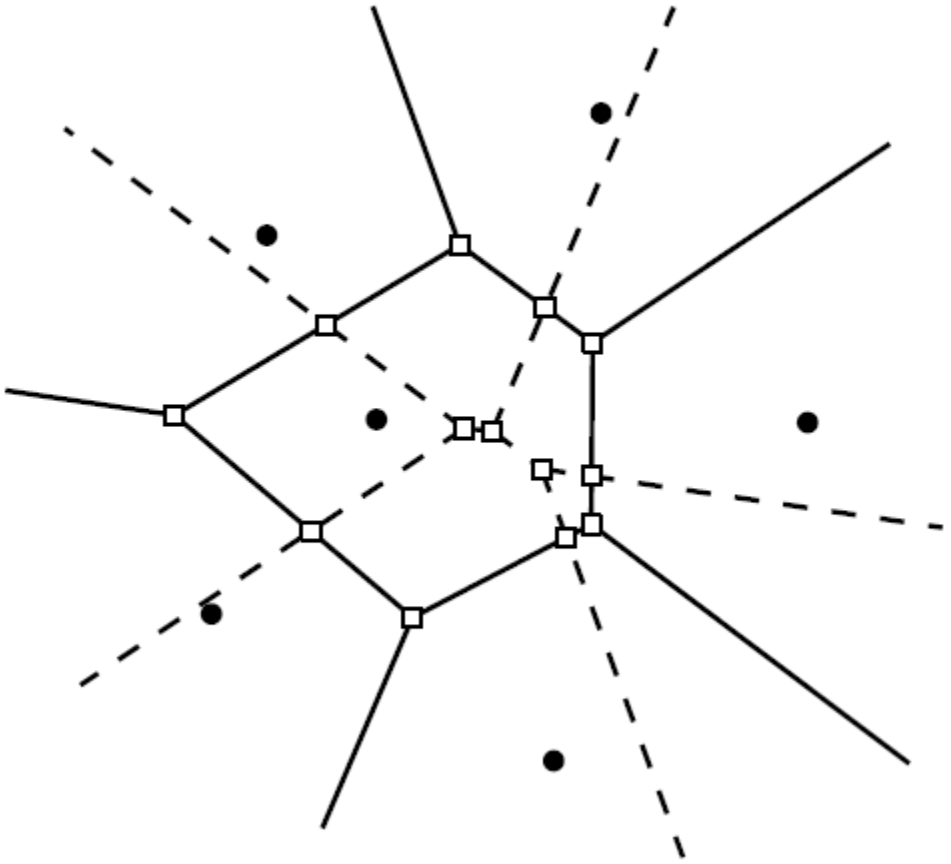
Then the two points on C_{inner} define a set of empty circles and the two points of C_{outer} define a set of full circles

the center of circle must lie on an edge

Of the voronoi diagram and farthest

Point voroni daigram.

1.The vertices of overlay
Are exactly the candidate
Centers of the smallest
Width annulus



2.For every pair of edege, one from

Each of the diagram , test if intersect.

Then candidate for 3 case.

And can be determinate in

$O(n^2)$time.

In case 1 and 2

1.Compute voronoi diagram and
farthest-point Voronoi diagram.

2.For each vertex of the FV(p), determine
The point of p that is closest
.
3.For each vertex in VP(P), determine
the point of p that is farthest.
This give us O(n) sets of four
Points that define the candidate
Annulus for case 1 and 2.

Theorem: The roundness, or the smallest-width annulus of n points in the plane can be determined in $O(n^2)$ time

END