CAD & Computational Geometry

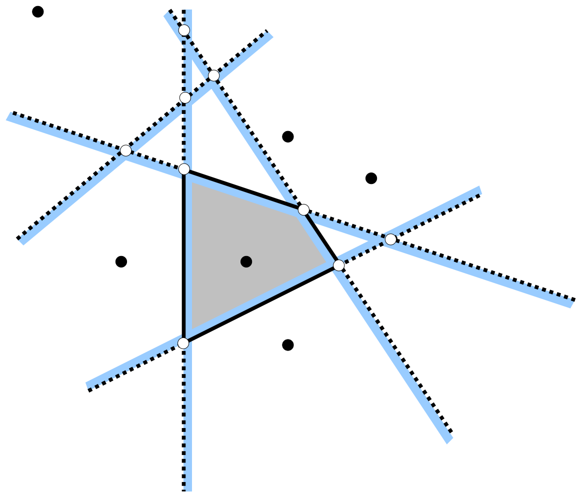
Course plan

- Introduction
- Segment-Segment intersections
- Polygon Triangulation
- Intro to Voronoï Diagrams & Delaunay Triangulations
- Geometric Search
- Sweeping algorithm for Voronoï Diagrams

Voronoi Diagrams

- How to compute the Voronoi diagram ?
  - For each cell $V(p_i)$, compute the intersections of all the half-planes $h(p_i,p_j)$ with $j \neq i$ using the intersection of lines of chapter CG2

    Complexity : in $n \log n$ for each cell (one does not know in advance which intersections will be found in the final shape of the cell)
  - There are n cells → $n^2 \log n$ globally. However, the complexity of the diagram is only $O(n)$...
  - Is it possible to be faster ? → yes !
    - Optimum : $\Omega(n \log n)$
    - Demonstration later on.



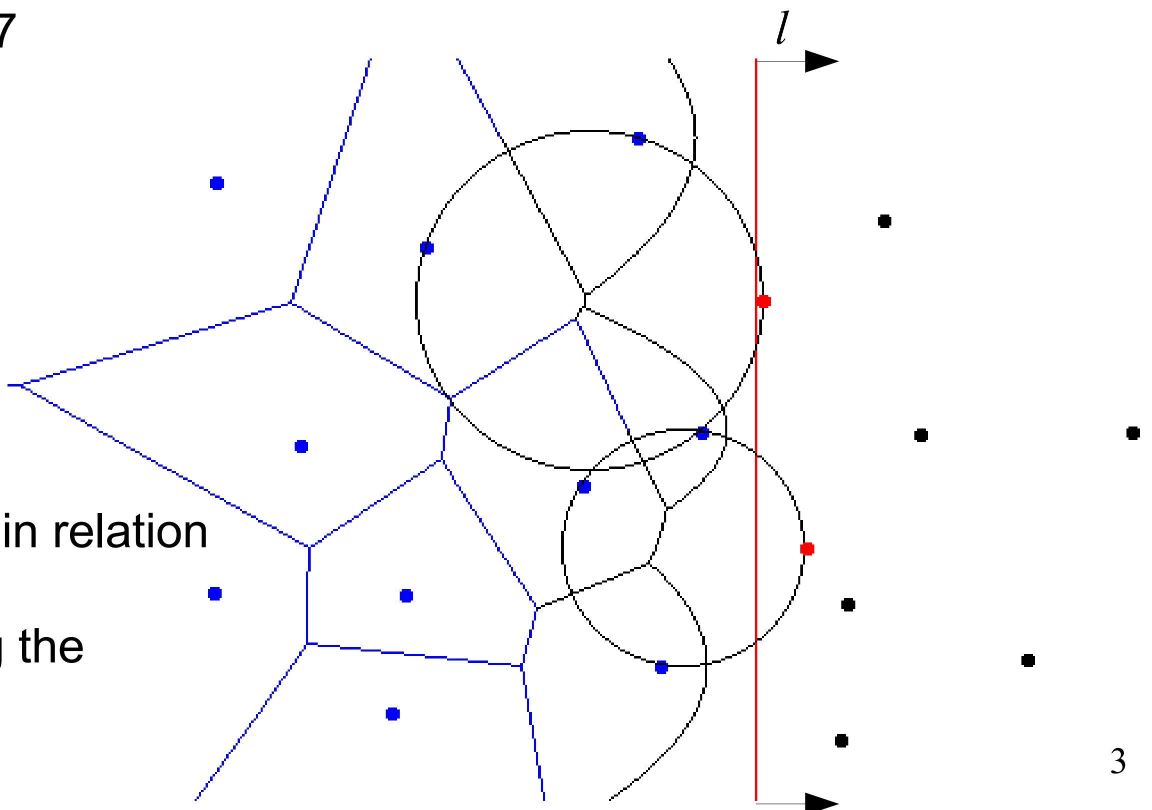
Voronoi Diagrams

- Line sweeping algorithme for the computation of a Voronoi Diagram
  - Also known as Fortune's algorithm

Steven Fortune, A sweepline algorithm for Voronoi diagrams, Algorithmica 2 (1-4),pp. 153-174, 1987
DOI: 10.1007/BF01840357

Algorithm in $O(n\log n)$

Known sweep line process :
- an imaginary line l sweeps the plane
- a status T is updated at each event
- the event list F contains event in the order in which they appear (x-wise)
- the Voronoi diagram is built step by step in relation with T
- By duality, it is another way of computing the Delaunay triangulation...

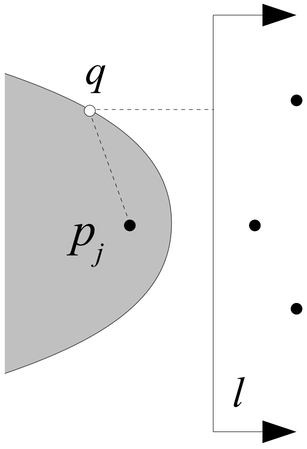


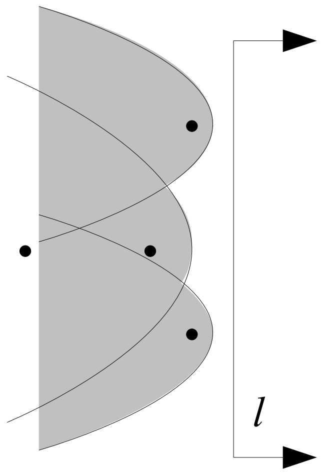
Voronoi Diagrams

- The paradigm of sweep line algorithms implies an invariant. Here, it will be the portion of the Voronoi diagram that we know will not change for subsequent events.
  - It is **not** the portion of the Voronoi diagram on the left of the line l.

    In fact, a part of $Vor(P)$ located on the left of l still depends on some of the points p_i that are located on the right.
  - More precisely, the part that does not change is the set of points q such that the distance to the line l is greater or equal to the distance to the poinst p_j located on the left of l.

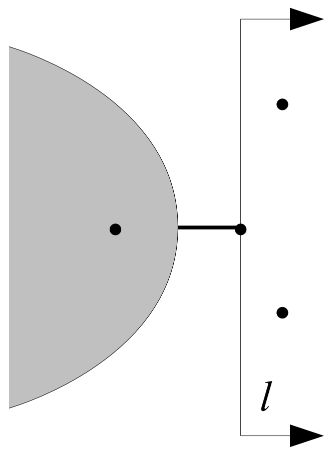
    The limit of such a domain is a succession of parabolas, shaped somewhat like a "beach line".
  - The "beach line" is y-monotone.
  - The status T will be this beach line.
    How does it change when events occur ?
    (What are in fact these events ?)

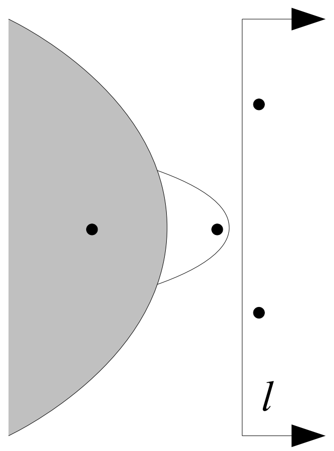




Voronoi Diagrams

- Configurational changes in the beach line
  - A new parabolic arc appears



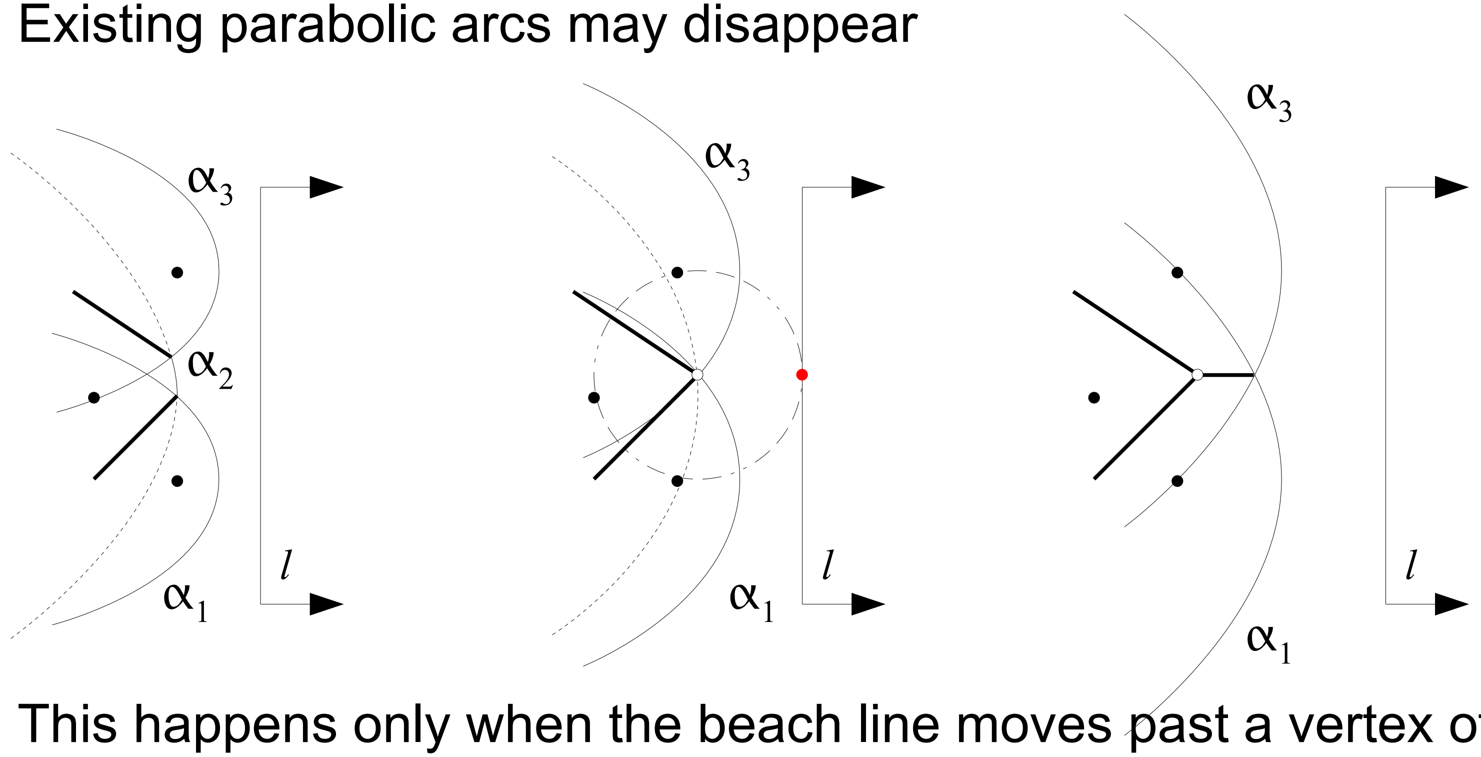


  - In fact, only when the line l sweeps past a point p_i. This is the first kind of events (points of P)

    Proof : cf Book , p 153

Voronoi Diagrams

- Configuration changes in the beach line
  - Existing parabolic arcs may disappear

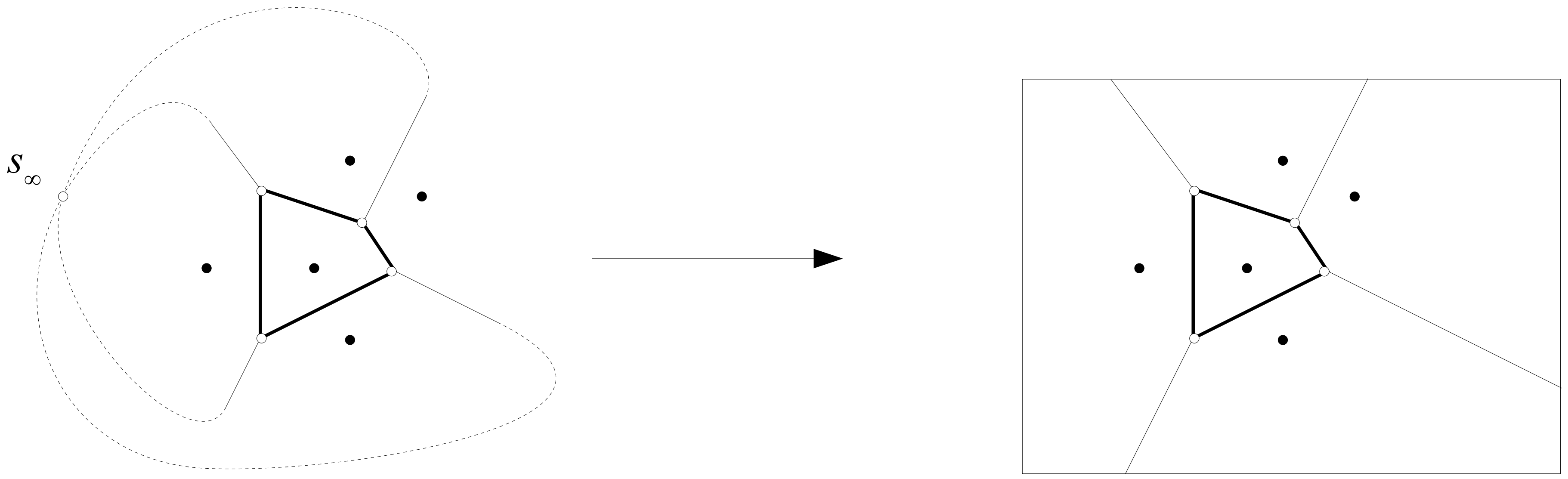


  - This happens only when the beach line moves past a vertex of the Voronoi diagram, otherwise said, when the line l meets the extremal point of the circle going through three points p_i that correspond the three consecutive arcs α_i of the beach line.
  - This is the second type of events (limit circles)

CAD & Computational Geometry

Voronoi Diagrams

- Data structures
  - The Voronoi diagram : e.g. a Doubly Connected Edge List. Ideally one needs a bounding box so that the cells are closed, and the half edges all able to have a geometrical meaning (a direction and starting point)



CAD & Computational Geometry

Voronoi Diagrams

- Data structures
  - Status T : The beach line – y-monotone – is a binary balanced tree. **The leaves** store the parabolic arcs and are sorted by increasing y coodinate. For each leaf, there is a pointer to an event in the even list – the limit circle event that correspond to the disapearing of the corresponding parabolic arc. If it does not yet exist, store e.g. a null pointer.

    The **internal nodes** inside the tree are the “breaks” in the beach line. Those are indentified with an ordered tuple (p_i, p_j) , where p_i is the point corresponding to the inferior parabola, and p_j that of the superior parabola. At each interior node, one ca store a pointer to a half-edge of the Voronoi diagram
  - One may look for the are immediately to the left of a given point p_k in$O(\log n)$. Upon tree traversal, at each internal node, it is sufficient to compare the y coordinate of the point p_k with the y coordinate of the “break”, which may be computed “on the fly” knowing the position of l (which goes through p_k).

Voronoi Diagrams

- Data structures
  - Event list F : It is a priority queue for which the events are sorted lexicographically by increasing x , then increasing y .
  - In this list, on can find all the points $p_i \in P$, and the limit circle events . The point used to classify the cicle events is the rightmost point of the circle. Moreover, a pointer to the arc inside the status T shall be stored.
  - Every point p_i is known in advance, however circle events are not and must be detected “on the fly”.

Voronoi Diagrams

- Determination of circle events
  - As each event is processed, triplets of contiguous arcs appear or disappear
    - Those correspond potentially to a circle event (in the future)
  - It is clear that at each new triplet of arcs, a circle event should be inserted in the list F.

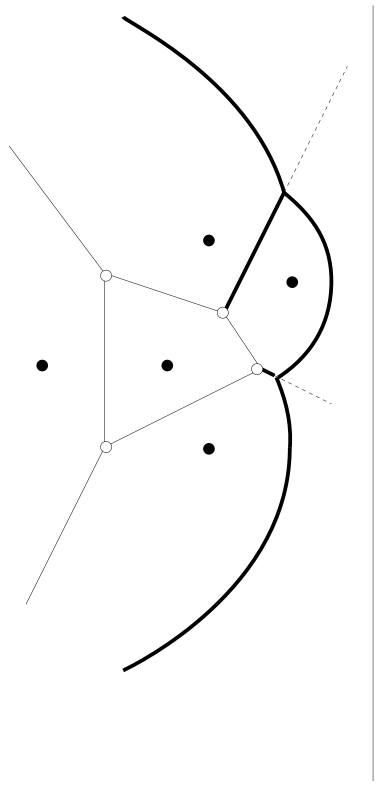
    Two remarks:
    - It is possible that, for a given triplet, the two edges that are concerned in the Voronoi diagram are divergent. In this case, no need to insert a new event (it would be in the past !)
    - If the edges are convergent it is still possible that the corresponding triplet disappear before it is processed. This would be because of another event (e.g. a point event), and has to be considered as a “false alarm”, thus the corresponding circle event should be deleted from L.

CAD & Computational Geometry

Voronoi Diagrams

Two remarks:

- It is possible that, for a given triplet, the two edges that are concerned in the Voronoi diagram are divergent. In this case, no need to insert a new event (it would be in the past !)



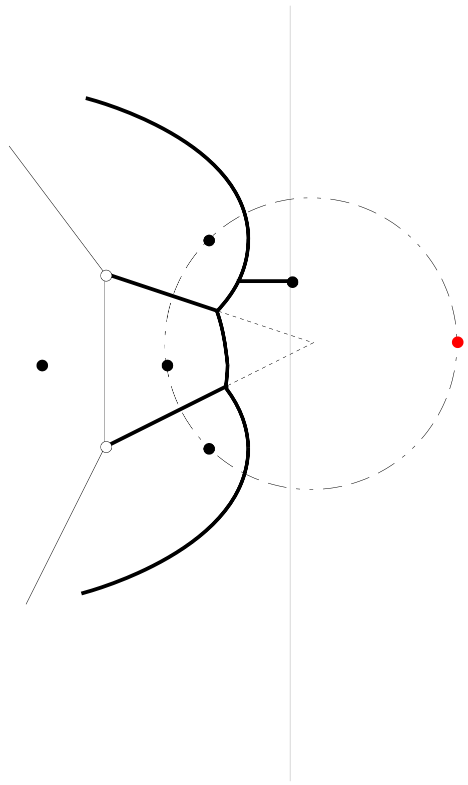
CAD & Computational Geometry

Voronoi Diagrams

Two remarks:

- If the edges are convergent it is still possible that the corresponding triplet disappear before it is processed. This would be because of another event (e.g. a point event), and has to be considered as a "false alarm", thus the corresponding circle event should be deleted from L.

  It is easy if there is a link between the arcs in the status and the circle events (a pointer) stored along with the arc in the status.

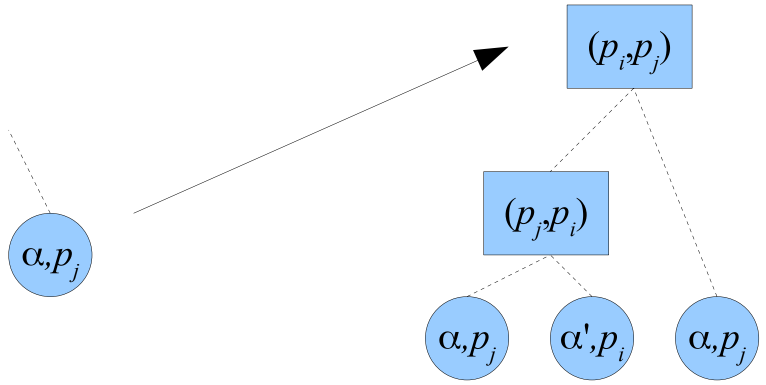


Voronoi Diagrams

- Global algorithm

```
Voronoi(P)
Input : a set of points $P=\{p_0,p_1 \dots , p_{n-1}\}$
Output : a DCEL of the V.D. of P in its bounding box (V)
{
  Initialize an event list $F$ with the points in  $P$ ( 'point' events)
  Initialize en empty status $T$
  Initializer an empty DCEL $V$
  While F contains events
  {
    Take the first event from F (lexicographic order)
    If it is a "point" event then ProcessPointEvt(pi)
    Else ProcessCircleEvt(f) // f is the leaf in the status corresponding to the arc that will be deleted
  }
  The internal nodes of  T are half edges of the V.D. they should be limited by the bounding box so that "outer" cells are bounded and allow traversal
  Walk the DCEL and create faces (cells)
}
```

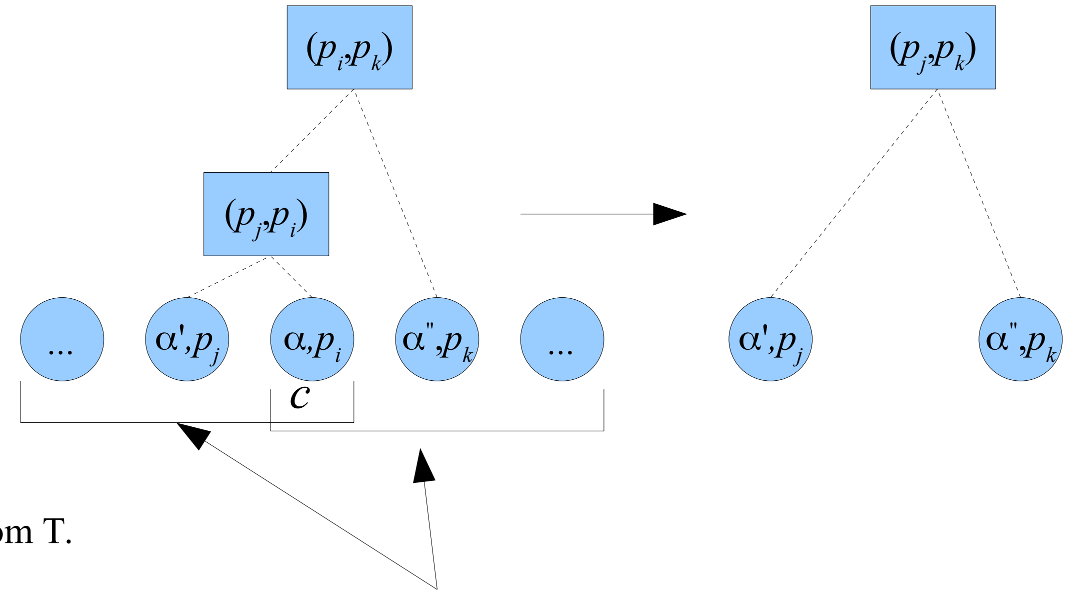
Voronoi Diagrams



```
ProcessPointEvt($p_i$)
{
  If T is empty, insert $p_i$ into T
  Else
  {
    Look for the arc $\alpha$ immediately on the left to $p_i$ ; it is associated to another point $p_j$
    If the leaf containing $\alpha$ has a pointer toward a circle event in $F$, delete that event.
    Replace the leaf in T with a subtree having 3 leaves  that correspond to the 3 new triplets
      In this subtree :
        The middle leaf corresponds to the new arc $\alpha_i$ associated to  $p_i$ ;
        The two others are copies of the old arc  $\alpha$ (cut in two)
        Insert two new internal nodes $(p_i,p_j)$ and $(p_j,p_i)$
    Re-balance the tree.
    Create two new half edges in the DCEL V, for the edge separating $V(p_i)$ et $V(p_j)$
    Check that the triplets of consecutive arcs that are above or below the new arc $\alpha_i$ (from  $p_i$ ) correspond to
converging edges : if yes then add a corresponding circle event in the list $F$.
  }
}
```

CAD & Computational Geometry

Voronoi Diagrams



```
ProcessCircleEvt(c)
{
  Delete the leaf c (that contain the arc α) from T.
  Update the tuples in the internal nodes of T
  Delete every circle event associated to α (they can be found by walking T from c)
  Add the center of the circle as a new vertex into the DCEL of the V.D.
  Create two new half edges that correspond to the new triplets
  Check that the new triplets correspond to converging edges : if yes then add a corresponding circle event in the list
F.
}
```

Voronoi Diagrams

- Some degenerate cases :
  - At the beginning of the algorithm : points having the same x coordinate
  - Concyclic points (4 or more)
  - Point event exactly on the right of a break in the beack line (same y coordinate)
  - Points corresponding to three successive arcs that are also co-linear

Some othe types of Voronoï diagrams

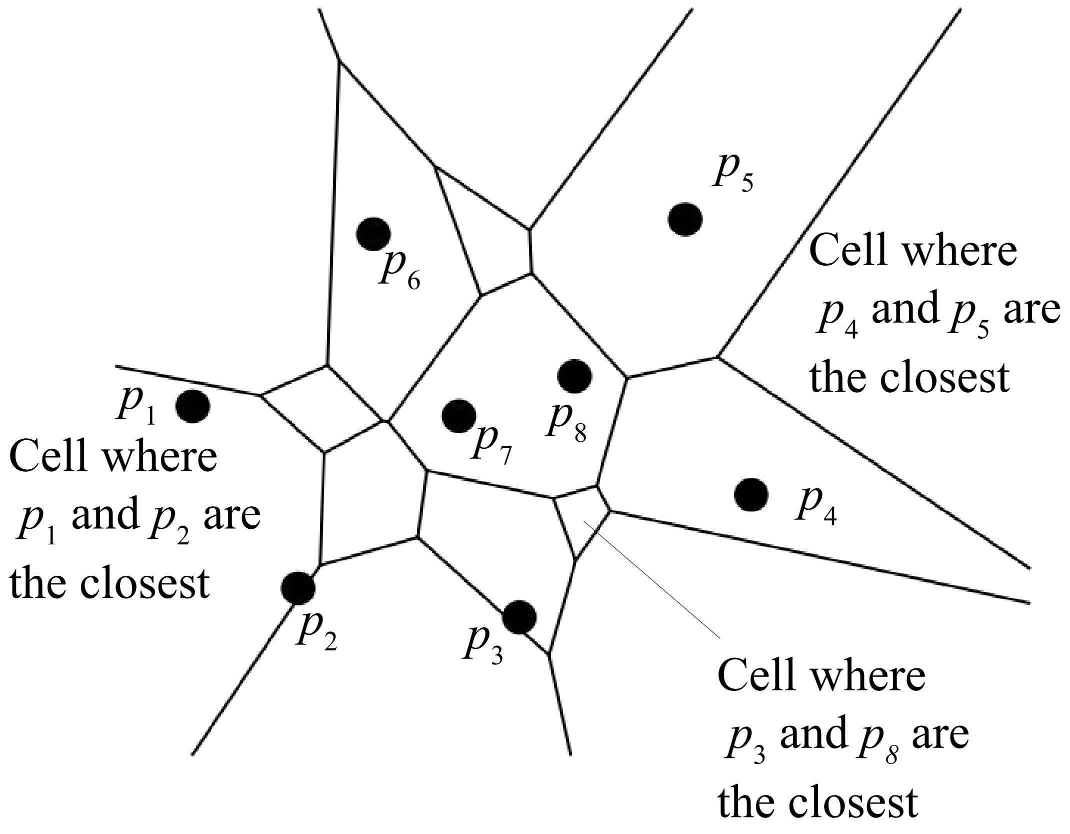
Voronoi Diagrams

- Order p Voronoi diagrams
  - Aims to build regions closest to p of the sites
  - Order 1 diagram is the regular Voronoi diagram
  - With n sites, it is possible to build a diagram of maximum order $n-1$
  - This diagram of order $n-1$ is also called the *Maximum Distance Diagram* (MDD)

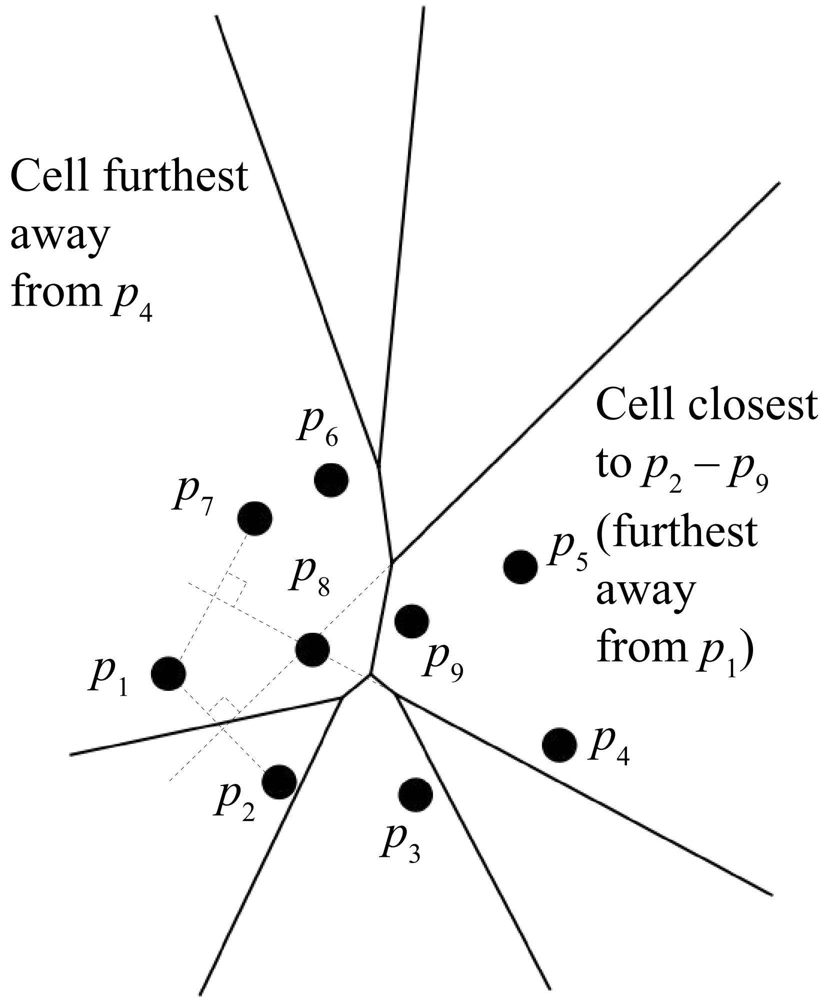
CAD & Computational Geometry

Voronoi Diagrams

- Order 2 diagram


p_5
p_6
Cell where p_4 and p_5 are the closest
p_8
p_1
p_7
Cell where p_1 and p_2 are the closest
p_4
p_2
p_3
Cell where p_3 and p_8 are the closest

- Order $n-1$ diagram

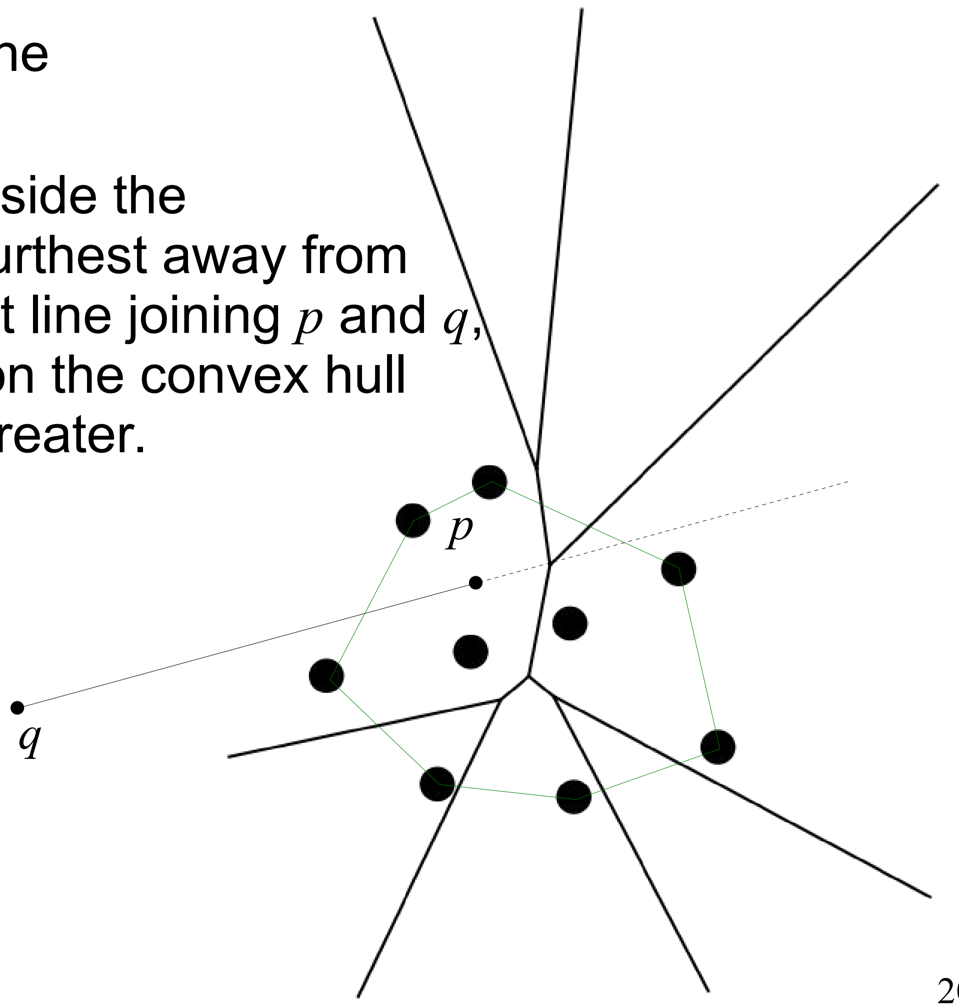

Cell furthest away from p_4
Cell closest to $p_2 - p_9$ (furthest away from p_1)
p_6
p_7
p_8
p_5
p_9
p_1
p_4
p_2
p_3

Voronoi Diagrams

- Properties of the MDD

  Let $P = \{p_1 \ldots p_{n-1}\}$ a set of points in the plane

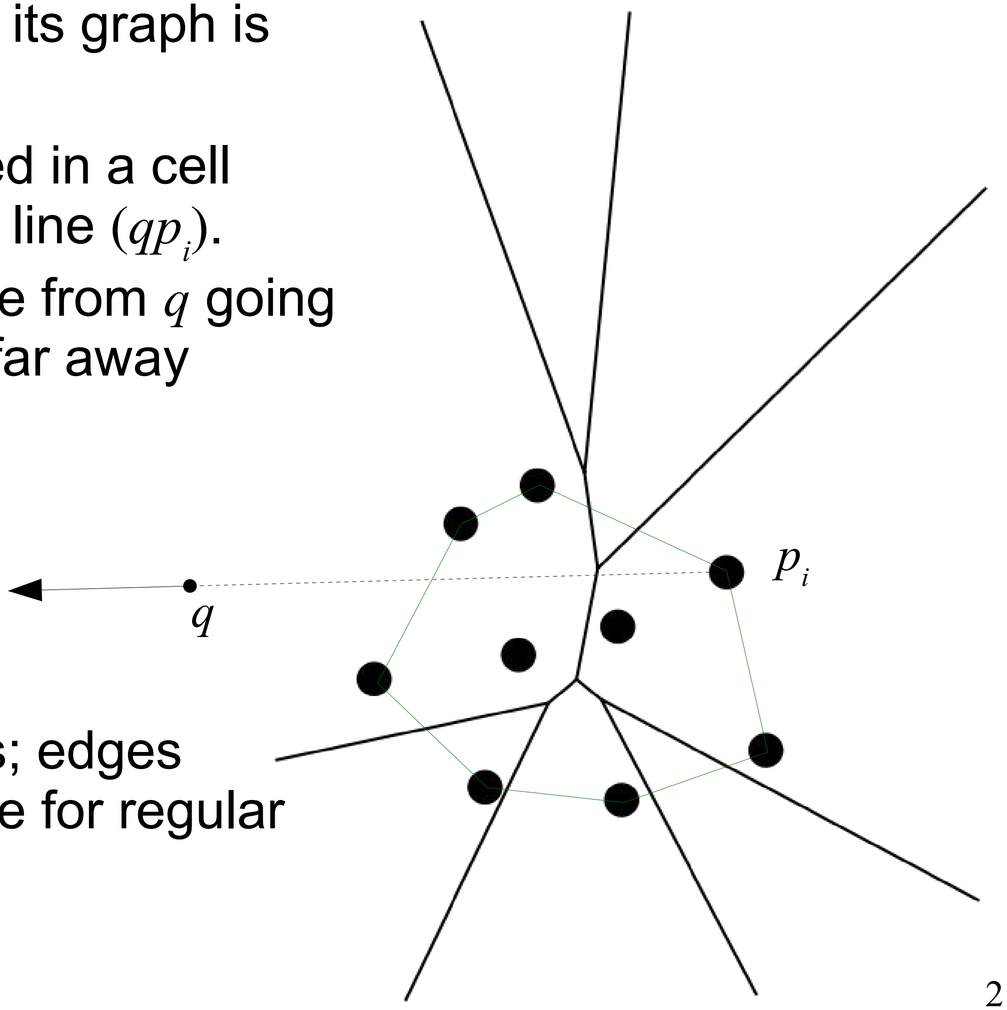
  - A point p_i of P is associated to a cell of the MDD of P only if it belongs to the convex hull of P.
  - The proof is trivial : a point p inside the convex hull of P, is never the furthest away from **any** other point q : on a straight line joining p and q, there necessarily exist points on the convex hull such that the distance to q is greater.



Voronoi Diagrams

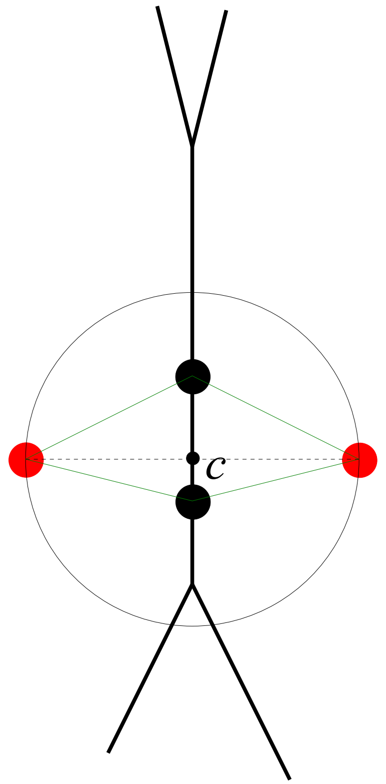
- Properties of the MDD
  - The cells are infinite, hence there are no cycles in the MDD, therefore its graph is a tree.

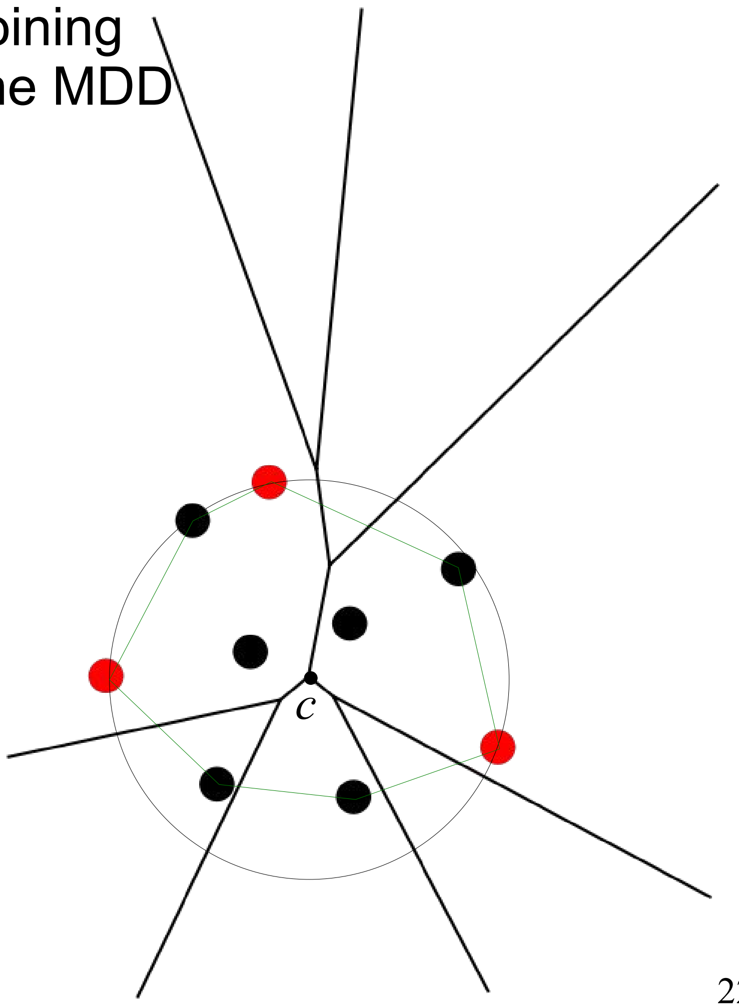
    Proof : from a point q ; located in a cell associated to p_i, one draw a line (qp_i). For every point of the half line from q going opposite to p_i, p_i is the most far away point from P.
  - There are $O(n)$ cells, vertices; edges (the proof is similar to the one for regular Voronoï diagrams) In particular, $n_e \leq 2n-3$



Voronoi Diagrams

- Properties of the MDD
  - The center c of any circumscribed circle to P (three points or more) is located on a vertex of the MDD
  - Or exactly on the middle of a segment joining the two sites associated to an edge of the MDD (two points).





Voronoi Diagrams

- Computation of the MDD

  - Computation of the convex hull of P. The vertices located on the convex hull are ordered in a cycle $s_0 \dots s_{h-1}$ (seen in a previous chapter)

  - One « mixes » randomly the vertices that hare located on the convex hull. Lets call $p_0 \dots p_{h-1}$ this ordering.

  - One takes the vertices one by one : k=0... h–4 ($p_{h-1} \dots p_3$)

  - The preceding vertex is stored (clockwise order) and the successor (anticlockwise) : to p_{h-k-1} correspond s_j, and we associate $s_{j-1 \bmod (h-k)}$ and $s_{j+1 \bmod (h-k)}$

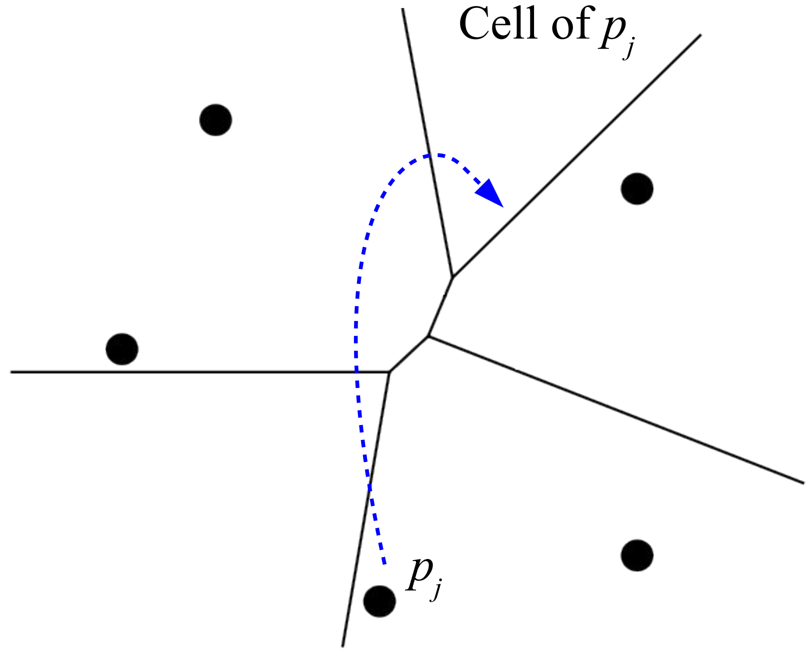
  - One takes off p_{h-k-1} (hence s_j) from the cycle (while updating the cycle so that it remains ordered …). Therefore, p_{h-k-1} cannot be the predecessor or successor of points canceled later on

  - Finally, one obtains a structure that allows ; starting from 4 points of the convex hull, to rebuild the convex hull while maintaining the ordering at every moment.

Voronoi Diagrams

- Computation of the MDD

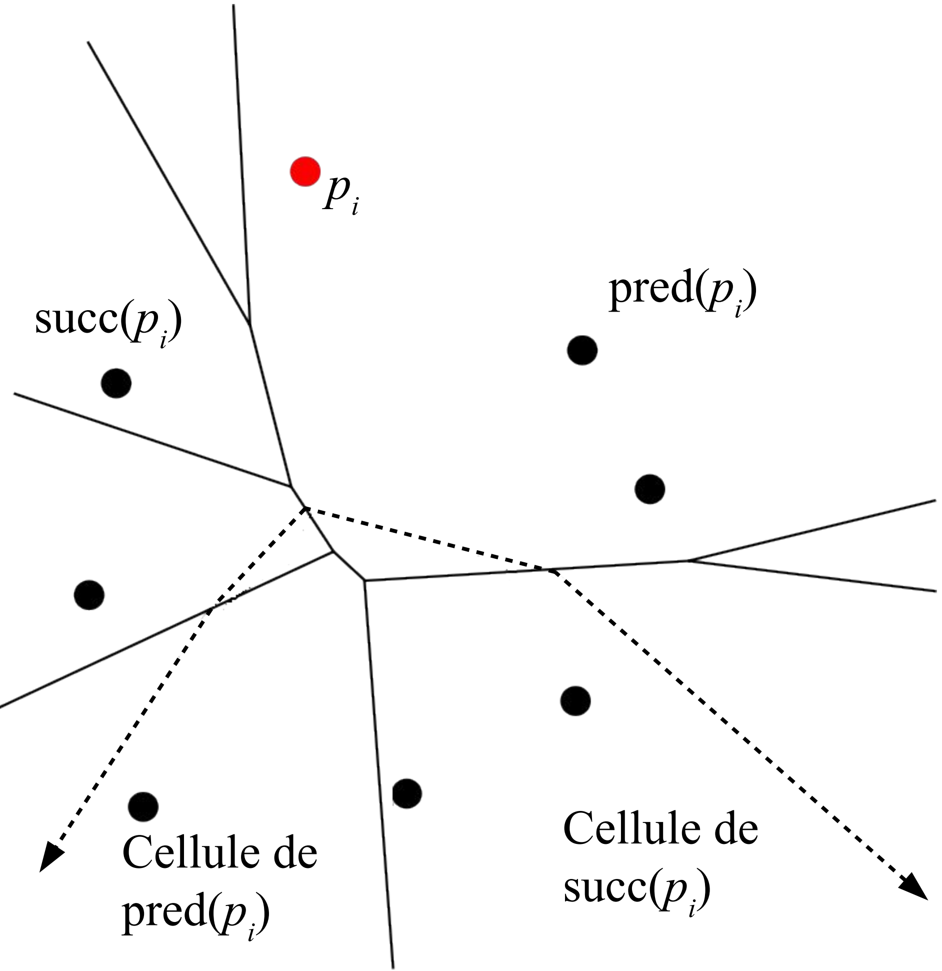
  - Associated to a point p_j, one keeps a pointer to the most anti-clockwise edge of the cell associated to p_j.

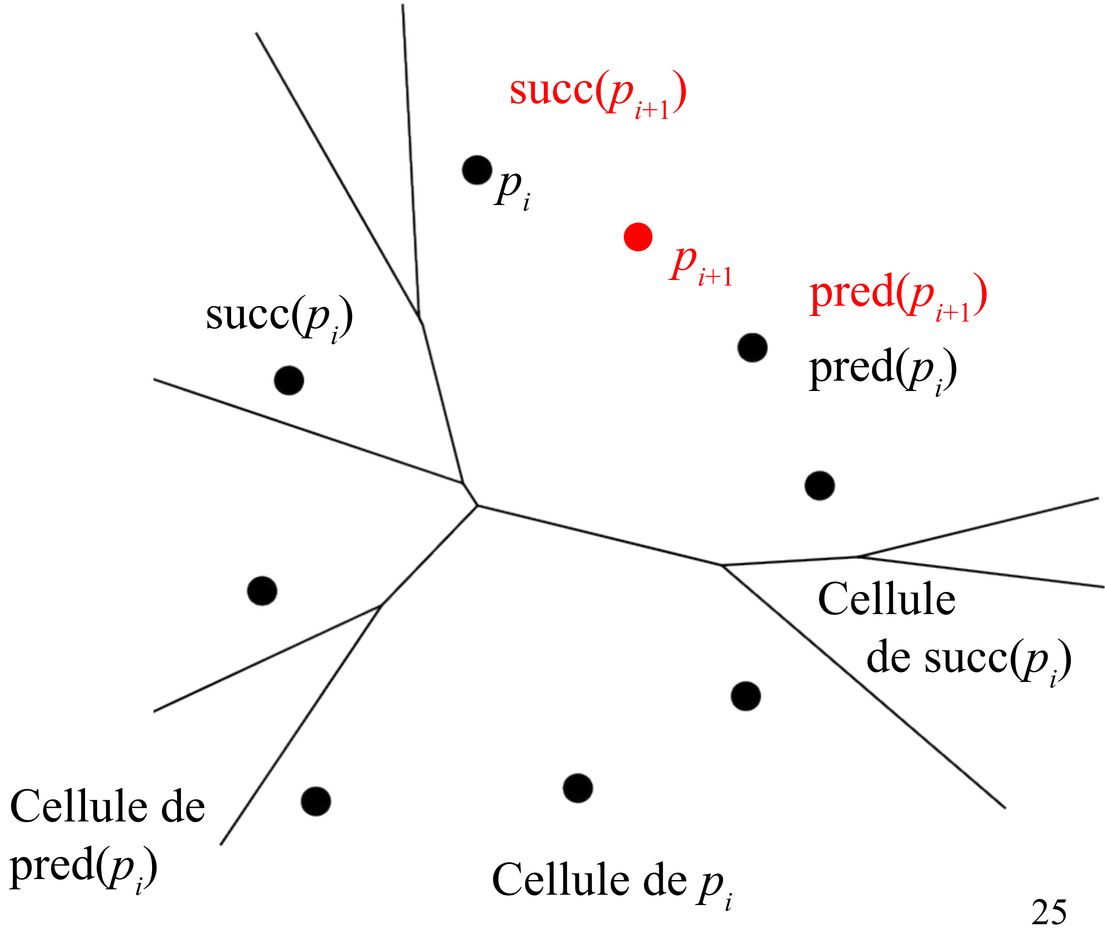


Voronoi Diagrams

- Computation of the MDD

  - Incrémental approach : from the MDD of p_{i-1} , one builds the MDD for p_i





Voronoi Diagrams

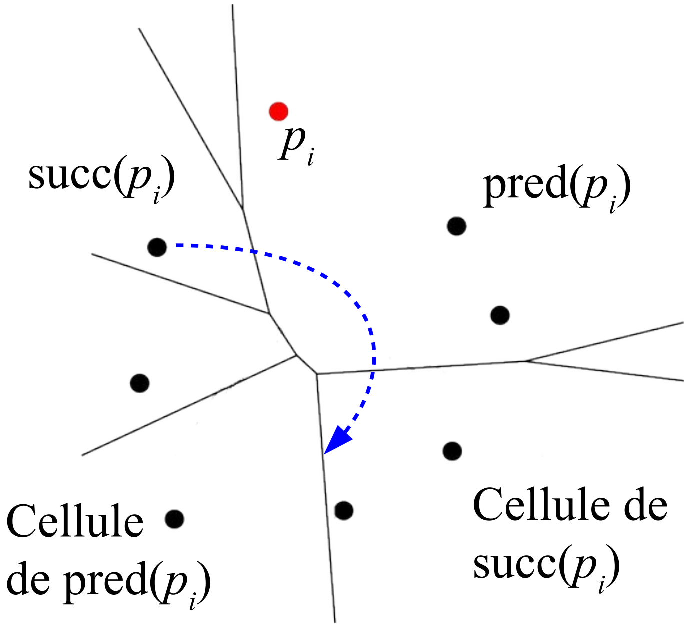
- Computation of the MDD

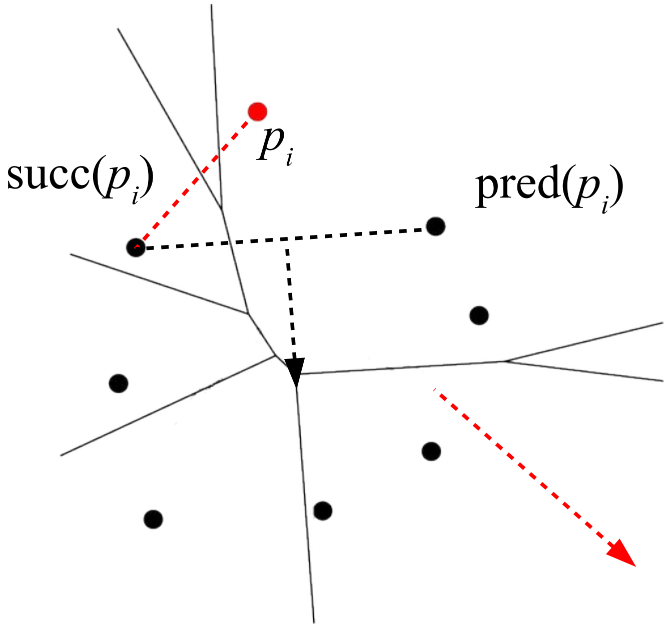
  - The cell for p_i comes “in between” that of $\mathrm{pred}(p_i)$ and $\mathrm{succ}(p_i)$

  - These two cells are separated by a half line that is part of the bissector of $\mathrm{pred}(p_i)$ and $\mathrm{succ}(p_i)$

  - $\mathrm{succ}(p_i)$ has in fact a pointer on this edge.

- - The bissector of p_i and $\mathrm{succ}(p_i)$ induces a new edge (a half line, in red)





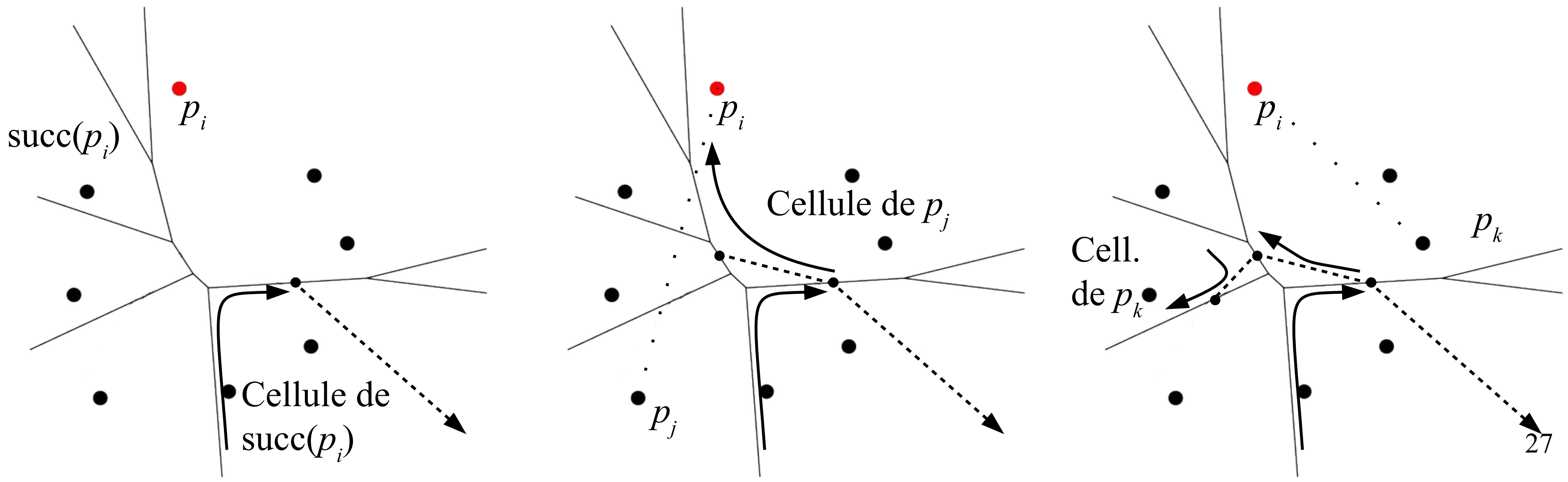
Voronoi Diagrams

- Computation of the MDD

  - One has to traverse the cell of succ(p_i) in the clockwise order to know which edge is intersected by the new half edge.

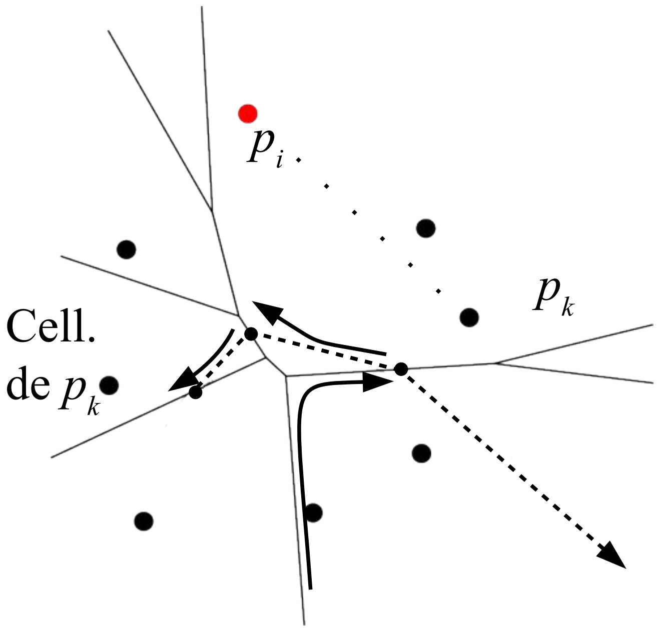
  - From the other side of this edge, there is the cell corresponding to another point p_j

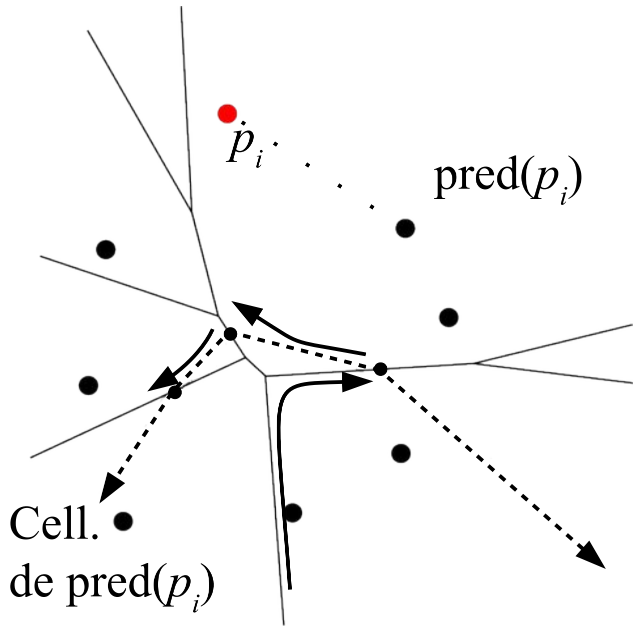
  - One traverse this cell again, to find the intersection with the bissector of p_i-p_j

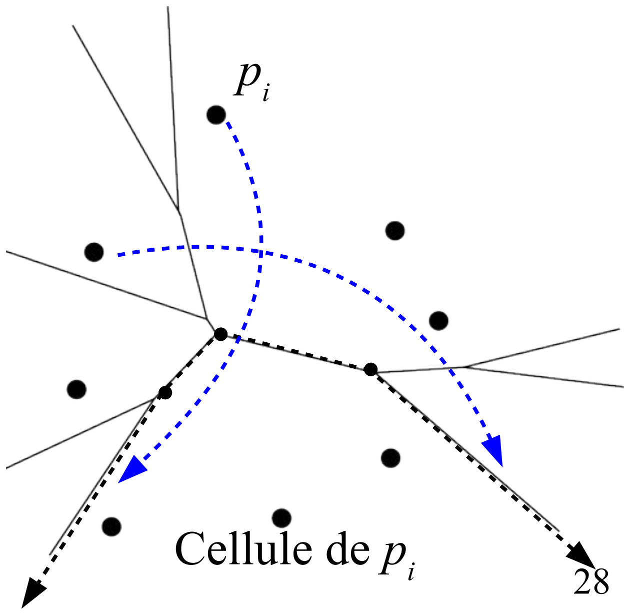


Voronoi Diagrams

- Computation of the MDD
  - - One repeats this set of operations until we meet pred(p_i), which wil give the other half-line that bounds the cell of p_i.
  - Do not forget to update the pointers between the existing points p_i and the corresponding edges (most anticlockwise) ...

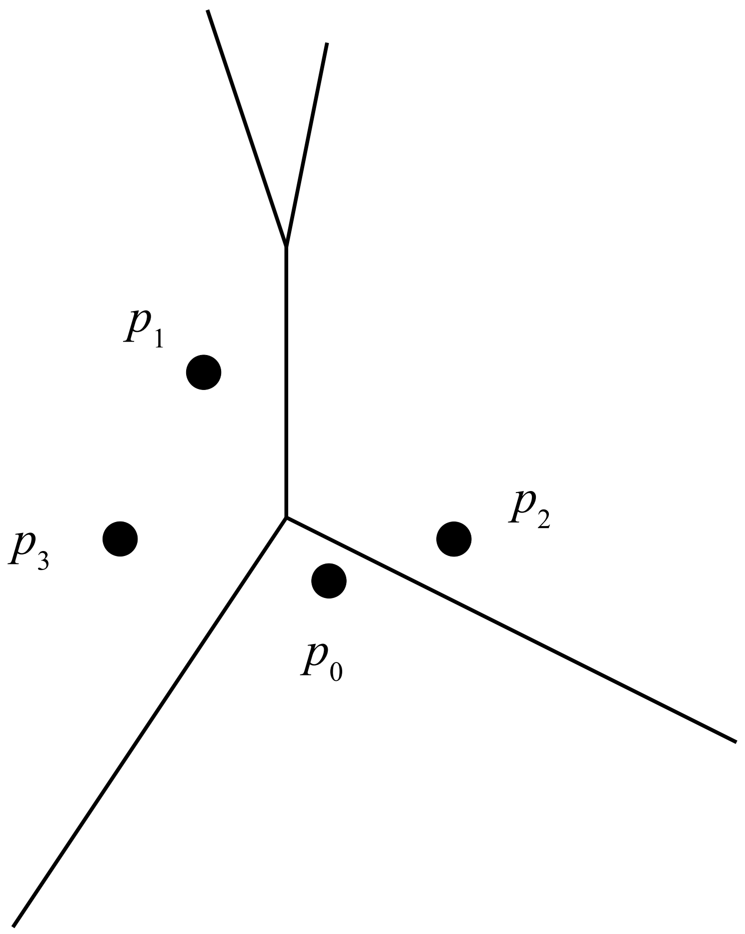






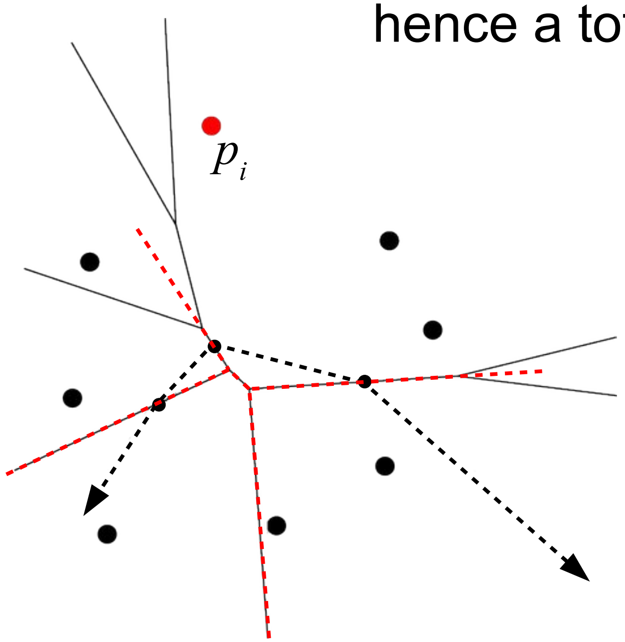
Voronoi Diagrams

- Computation of the MDD
  - The computation may be initialized with a MDD of the 4 first vertices $p_0 .. p_3$ for which the MDD is trivial :



Voronoi Diagrams

- Complexity of the algorithm
  - Initial convex hull : $O(n\log n)$, contains the n points in the worst case.
  - After the insertion of p_i, the corresponding cell has k edges.
  - Hence, to build this cell from the MDD obtained after the insertion of p_{i-1}, the algorithm has walked in k neighboring cells.
  - These k cells are separated by a subtree that has at most $2k–3$ distinct edges, and in the worst case, one will walk these edges two times, hence a total of $4k–6$ tests (linear in k)

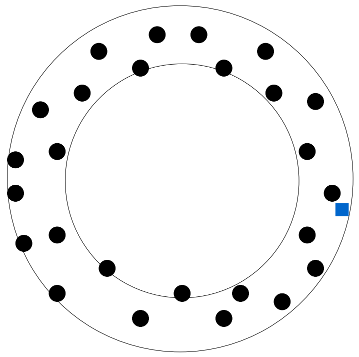


- The MDD at step i has at most $2i–3$ edges. Each edge separates two cells. So, in average, each cell has less than $(4i–6)/i$ edges, which is smaller than 4.
- Each point p_i has the same probability to be inserted at the end of the process, so globally the complexity to build the MDD is in $O(n)$.
- Globally, the complexity is therefore in $O(n\log n)$ with a storage in $O(n)$.

CAD & Computational Geometry

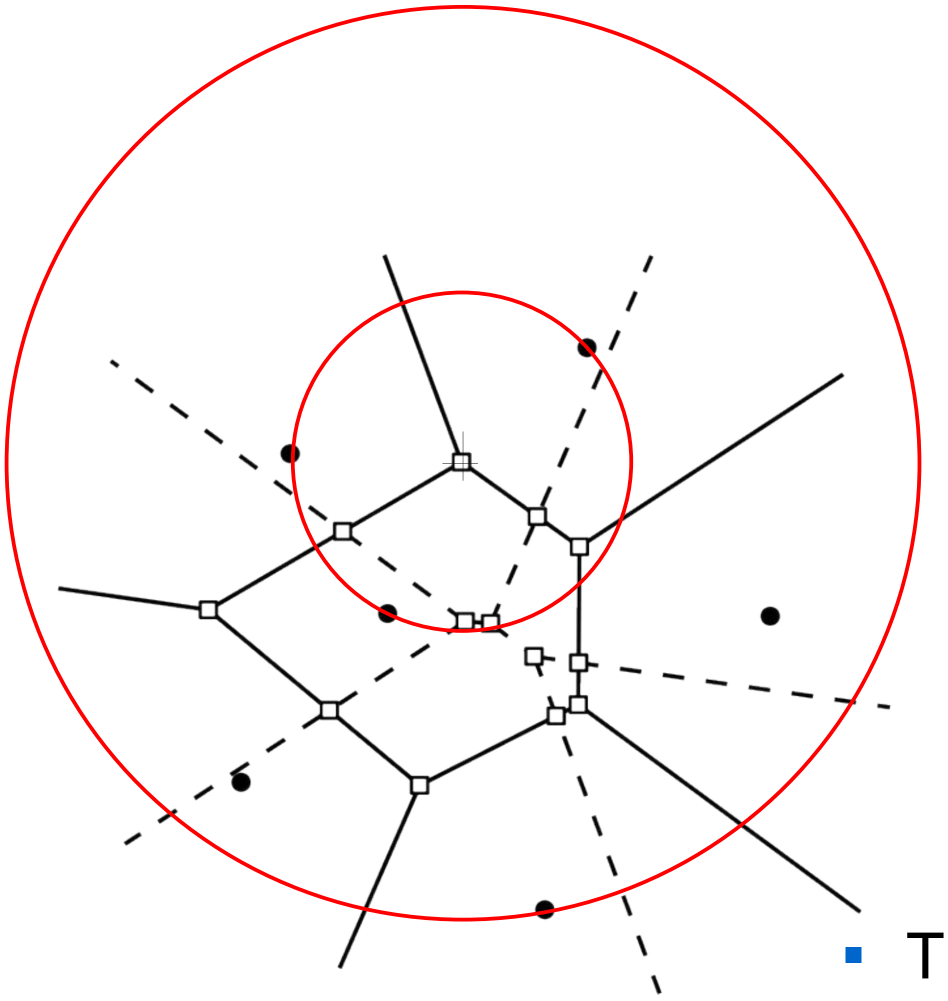
Voronoi Diagrams

- Application of the MDD
  - Metrology (in relation with the classical V.D.)
    - Dertermine the two min-max circles having the same center that approximate a cloud of samples (coming e.g. from a laser scanner operating on a machined cylindrical part, to asses the quality of the machining)
    - There are multiple cases :
      - The interior circle (max size) goes through 3 or more samples – its center is therefore on a vertex of the classical V.D. The exterior circle is then coincident with only one point (the one corresponding to a cell of the MDD)
      - The exterior circle (min size) goes trough three or more points – its center is therefore on a vertex of the MDD. The internal circle is then coincident with only one sample (that corresponding the a cell of the V.D.)
      - Finally; the external circle goes through two points, as well as the interior circle. The center is therefore at the intersection of an edge of the MDD, and one of the V.D.
    - To compute these circles and find the optimum, one must therefore compute the arrangements between a V.D. and the MDD.



CAD & Computational Geometry

Voronoi Diagrams

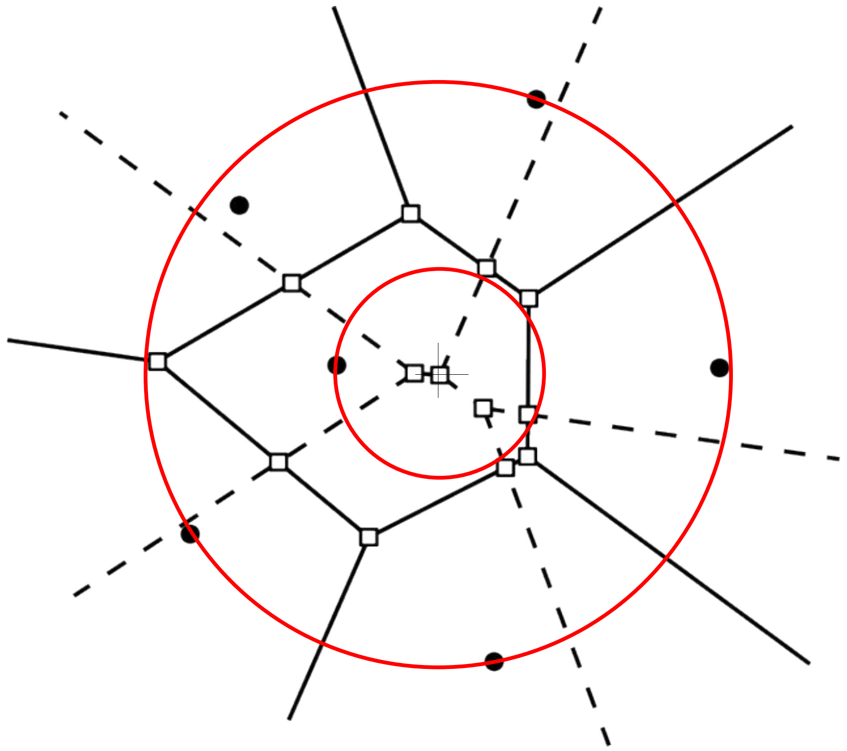


- For each vertex of the MDD, find the point of P that is the closest (gives 4 points)
- For each vertex of the V.D., find the point of P the most far away (gives 4 points)
- For each intersection between the V.D. and the MDD, determine the 4 points in P
- For each of these groups of candidates, find the one which gives the smallest ring.

- This may be done in $O(n^2)$ in the worst case
  - There may be $O(n^2)$ additional sites due to the intersection of the two graphs.

CAD & Computational Geometry

Voronoi Diagrams

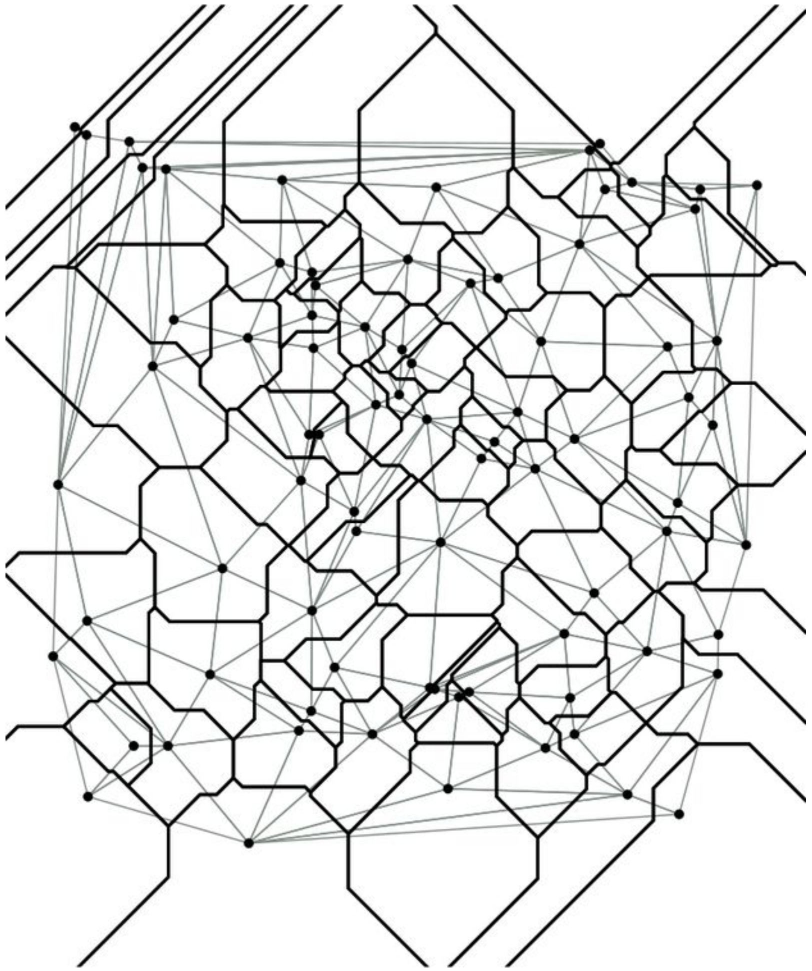


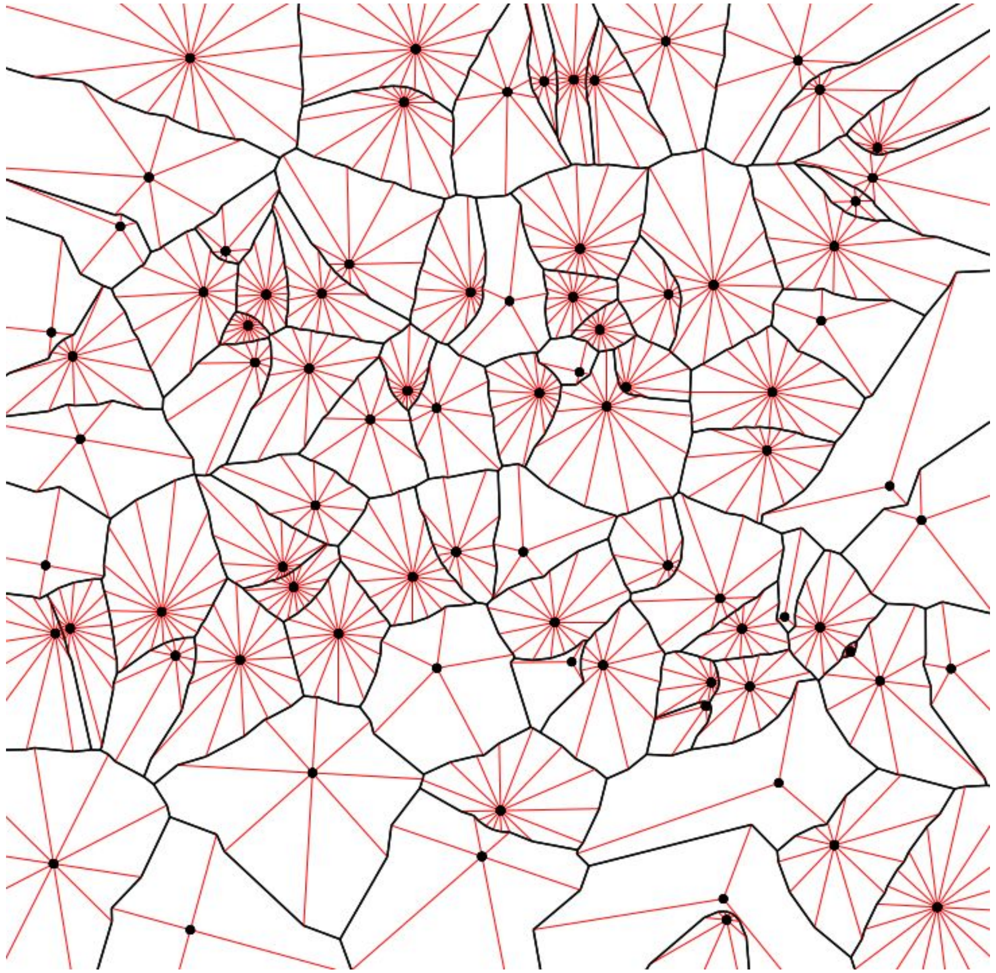
- In the present case; 13 combinations have to be checked

CAD & Computational Geometry

Voronoi Diagrams

- Voronoi diagrams in other norms ($L^p, p \neq 2$; polygonal ...)


Uniform L^1 norm


“local” polygonal norms