The Distributed Deployment of Mobile Sensors i.e. the Voronoi Diagram Construction Problem

Prof. Tiziana Calamoneri
Network Algorithms
A.y. 2017/18

The Problem

The Problem (1)

- We have already spoken about *mobile sensor networks*…
- … and of the *deployment problem.*
- A centralized solution is not always desirable because:
  - Connection with a server is required
  - Long delays are expected
  - The solution is not fault-tolerant
- The ability of moving around facilitates sensors to self-deploy starting from any initial configuration to a final distribution that guarantees that the AoI is completely covered.

The Problem (2)

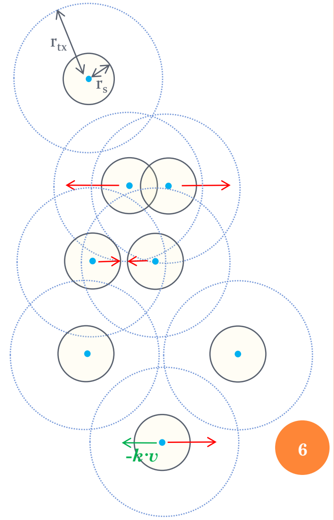
- The self-deployment is necessary in "hostile" environments:
  - Contaminated places
  - Fires
  - Battlefields…
- In these cases, sensors should position themselves and transmit the collected information.

The Problem (3)

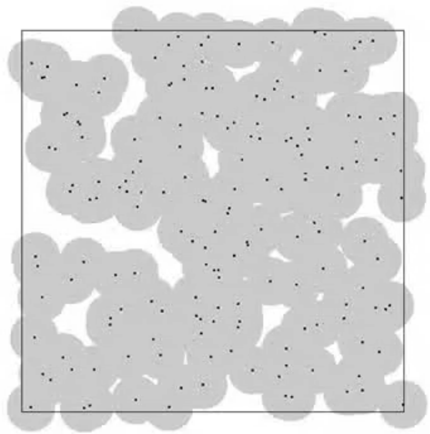
- Obs. The deployment problem is strictly related with the classical computational geometry problem called *art gallery* problem.
- In this problem the aim is to determine, in a polygonal environment, the minimum number of cameras necessary to guarantee that the whole room is supervised.
- There exist several algorithms to solve the art gallery problem, but all of them assume a perfect knowledge of the environment => another approach is necessary.

A Possible Approach: Virtual Forces (1)

- Idea: sensors are similar to charged particles (magnetic force) having a mass (gravitational force)
- Two sensors repel each other if they are too close
- Two sensors attract each other if they are far but can anyway communicate
- Two sensors ignore each other if they cannot communicate (too far)
- Friction to attenuate oscillations



A Possible Approach: Virtual Forces (2)



A Possible Approach: Virtual Forces (3)

Weaknesses:

- It is necessary a manual tuning of parameters
- Sensor oscillation
  - Friction forces
  - Stopping conditions
- In some versions, attracting effect of the border and of the obstacles (e.g. when only repulsive forces are considered)
- …

A Possible Approach: Virtual Forces (4)

Weaknesses (cntd):

- Sensors tend not to pass through doors and narrows

Possible lesson

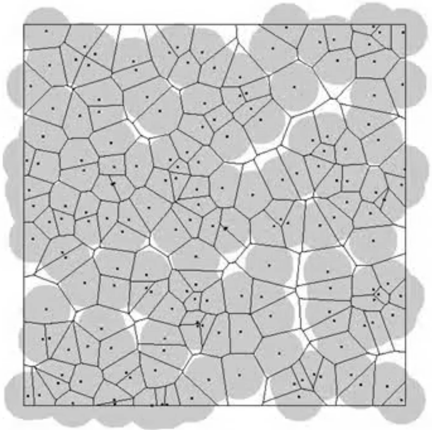
A Protocol Based on Voronoi Diagrams (1)

Idea:

- Each sensor is assigned an AoI portion and it has to take charge of it, trying to cover it as best as it can
- The sensor is "satisfied" if:
  - It completely cover its portion

  or
  - All its sensing radius is used to cover its portion
- If a sensor is not "satisfied" it has to move in order to improve its coverage
- AoI portions can be assigned according to the Voronoi diagram.

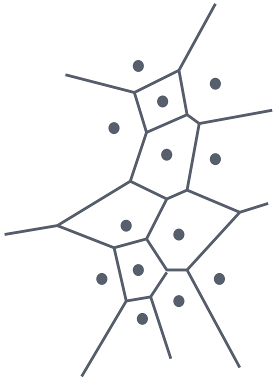
A Protocol Based on Voronoi Diagrams (2)



Voronoi Diagrams

Voronoi Diagram (1)

Suppose that you live in a desert where the only sources of water are a few springs scattered here and there. For each spring, you would like to determine the locations nearest that spring. The result could be a map, like the one shown here, in which the terrain is divided into regions of locations nearest the various springs.



Voronoi Diagram (2)

Maps like this appear frequently in various applications and under many names. To mathematicians, they are known as *Voronoi diagrams*.

Voronoi diagrams are rather natural constructions, and it seems that they, or something like them, have been in use for a long time.

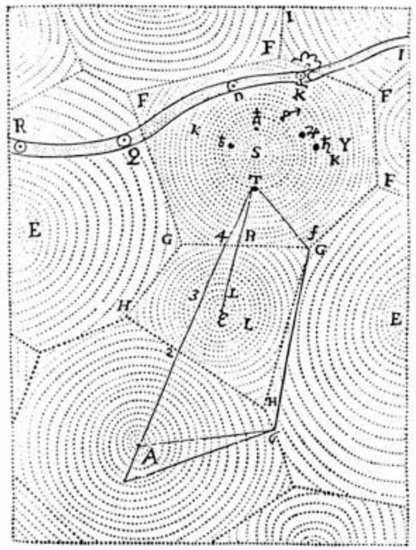
Voronoi Diagram (3)

Voronoi diagrams have been used by:

- anthropologists to describe regions of influence of different cultures;
- crystallographers to explain the structure of certain crystals and metals;
- ecologists to study competition between plants;
- economists to model markets in a certain economy;
- …

Voronoi Diagram (4)

- An informal study of Voronoi diagrams dates back to Descartes (1644): he includes the following figure with his demonstration of how matter is distributed throughout the solar system.
- …



Voronoi Diagram (5)

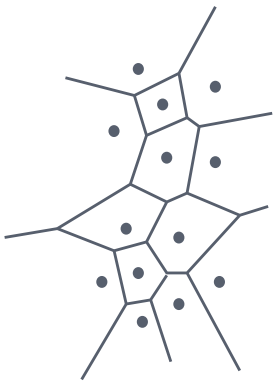
- …
- The English physicist Snow uses them for his analysis of the London cholera outbreak of 1854:
  - Snow considers the sources of drinking water, pumps distributed throughout the city, and draws a line labeled "Boundary of equal distance between Broad Street Pump and other Pumps," which essentially indicates the Broad Street Pump's Voronoi cell.
  - This map supports Snow's hypothesis that the cholera deaths are associated with contaminated water, in this case, from the Broad Street Pump. Snow recommends to the authorities that the pump handle be removed, after which action the cholera outbreak quickly ends.
- …

Voronoi Diagram (6)

- Dirichlet uses Voronoi diagrams in his studies on quadratic equations in 1850.
- Voronoi diagrams are so called in honor of the Russian mathematician Georgy F. Voronoi, who defined and studied them in the *n*-dimensional space in 1908.
- They are also called *Thiessen polygons* in meteorology in honor of the US meteorologist Alfred H. Thiessen; *Wigner-Seitz cells* in physics, *fundamental domains* in group theory and *fundamental polygons* in topology.

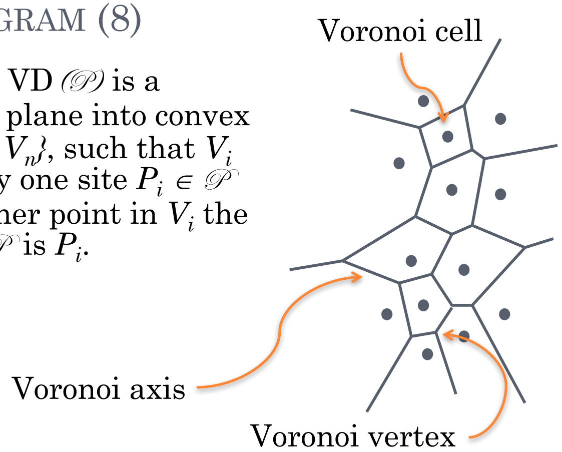
Voronoi Diagram (7)

- Def. of Voronoi Diagram:
  - $\mathscr{P}$: set of n distinct sites on the plane
  - VD $(\mathscr{P})$: partition of the plane into n cells V_i such that:
    - each V_i contains exactly one site
    - if a point Q on the plane is in V_i then $dist(Q, P_i) < dist(Q, P_j)$ for each $P_j \in \mathscr{P}, j \neq i$.



Voronoi Diagram (8)

- In other words: VD $(\mathscr{P})$ is a partition of the plane into convex regions $\{V_1, \ldots, V_n\}$, such that V_i contains exactly one site $P_i \in \mathscr{P}$ and for each other point in V_i the closest site in $\mathscr{P}$ is P_i.

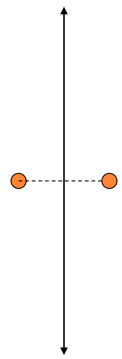


Voronoi Diagram (9)

Voronoi diagram of a single site

Voronoi Diagram (10)

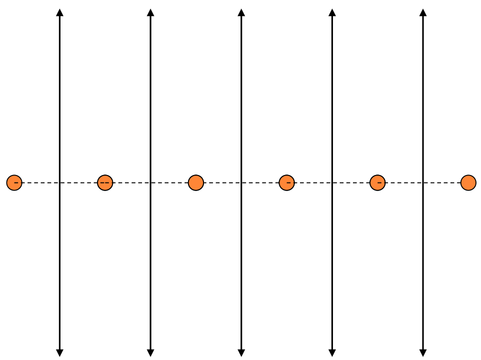
Voronoi diagram of two sites



The axis extends to infinity in both directions, generating two halfplanes

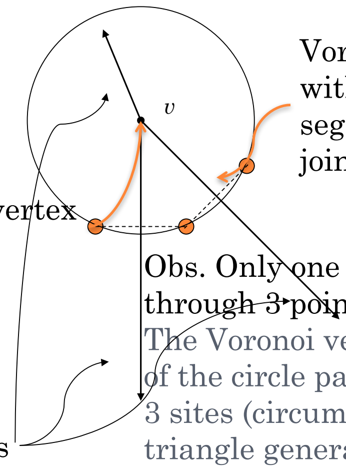
Voronoi Diagram (11)

Voronoi diagram of some colinear sites



Voronoi Diagram (12)

Voronoi diagram of 3 not colinear sites



Voronoi axes coincide with the axes of the segments pairwise joining the sites

The Voronoi vertex Has degree 3

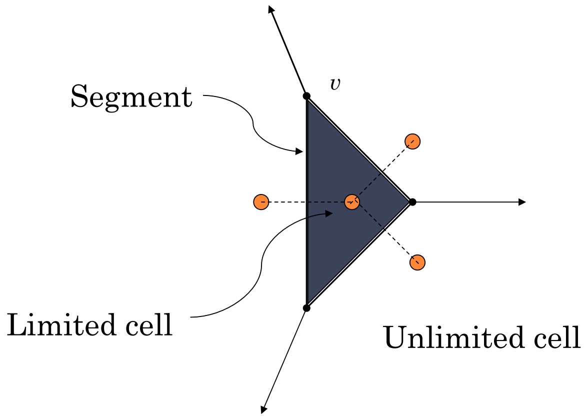
Obs. Only one circle passes through 3 points

The Voronoi vertex is the center of the circle passing through the 3 sites (circumscribed to the triangle generated by the 3 sites)

halflines

Voronoi Diagram (13)

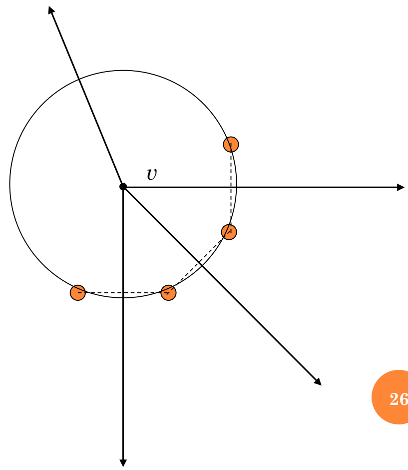
Voronoi diagram of 4 not colinear sites



Voronoi Diagram (14)

Not always 4 not colinear sites create a limited cell:

General position assumption: each 3 sites are not colinear and each 4 sites are not cocircular. Thanks to this assumption, all vertices have degree 3!

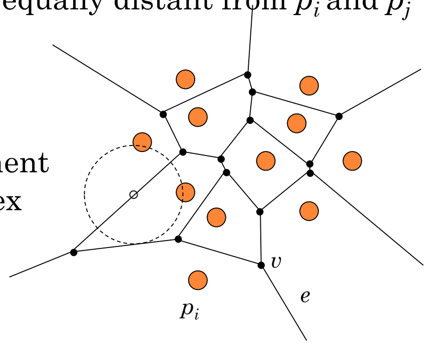


Voronoi Diagram Properties (1)

A point q on the plane lies on the Voronoi segment between p_i and p_j *iff* the largest empty circle centered in q touches only p_i and p_j.

– A Voronoi segment is a subset of a Voronoi axis, i.e. the set of point equally distant from p_i and p_j

p_i : sites of $\mathscr{P}$
e : Voronoi segment
v : Voronoi vertex

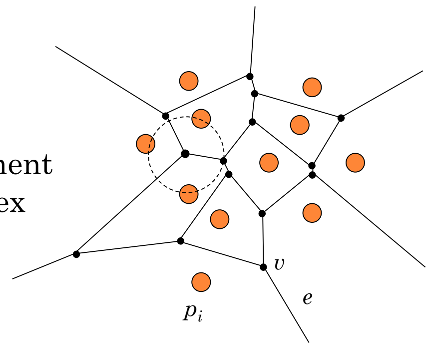


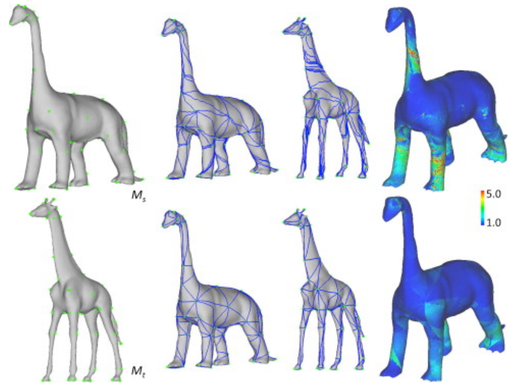
Voronoi Diagram Properties (2)

A point q in the plane is a Vornoi vertex *iff* the largest empty circle centered in q touches at least 3 sites of $\mathscr{P}$.

A Voronoi vertex is the intersection of at least 3 axes, each generated by a pair of sites.

p_i : sites of $\mathscr{P}$
e : Voronoi segment
v : Voronoi vertex

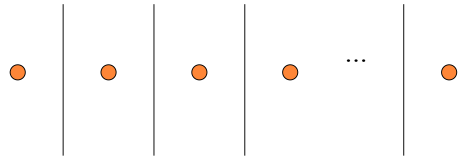




VORONOI DIAGRAM COMPLEXITY

VORONOI DIAGRAM COMPLEXITY (1)

- Th.: $|v| \leq 2n - 5$ and $|e| \leq 3n - 6$ for each $n \geq 3$.
- Proof: (Easy case)



Colinear sites → $|v| = 0$, $|e| = n - 1$

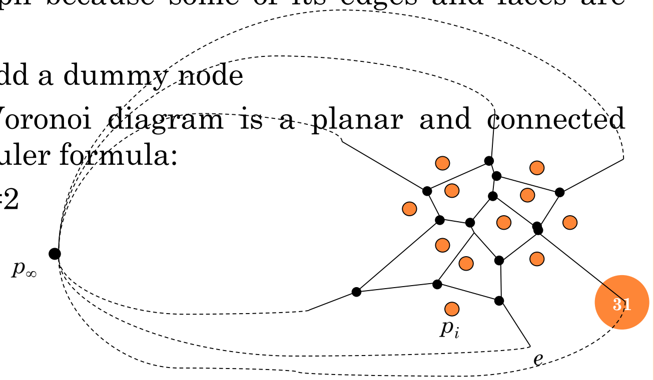
VORONOI DIAGRAM COMPLEXITY (2)

Proof of Th.: $|v| \leq 2n - 5$ and $|e| \leq 3n - 6$ for each $n \geq 3$ – cntd.

Proof: (General case)

- Problem: A Voronoi diagram cannot be considered as a planar graph because some of its edges and faces are unlimited
- Solution: add a dummy node
- Now the Voronoi diagram is a planar and connected graph → Euler formula:

  $|v| - |e| + f = 2$



VORONOI DIAGRAM COMPLEXITY (3)

Proof of Th.: $|v| \leq 2n - 5$ and $|e| \leq 3n - 6$ for each $n \geq 3$ – cntd.

$f = n + 1$. Euler formula becomes:

$|v| - |e| + n + 1 = 2$ (1)

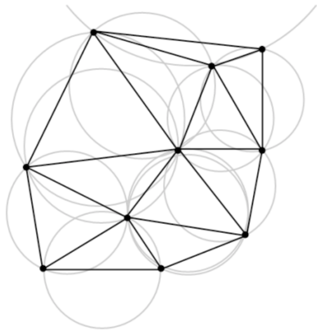
Moreover: $\sum_{v \in VD} \deg(v) = 2|e|$

since $\deg(v) \geq 3$ → $2|e| \geq 3|v|$ (2)

Joining (1) e (2):

$|v| \leq 2n - 5$

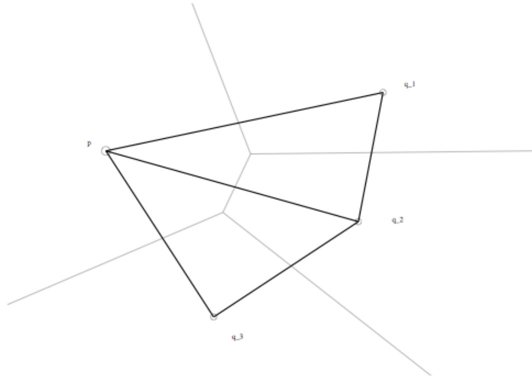
$|e| \leq 3n - 6$



The Dual Problem of the Voronoi Diagram

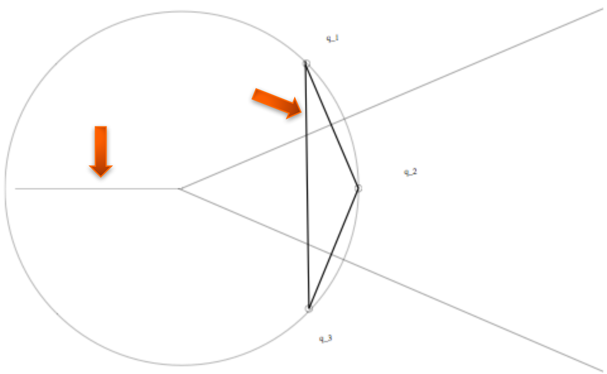
The Dual Problem of the Voronoi Diagr.

- The dual problem w.r.t. the decomposition of the plane into Voronoi cell is the Delaunay triangulation (obtained interescting each Voronoi axis with a segment joining the generating sites)



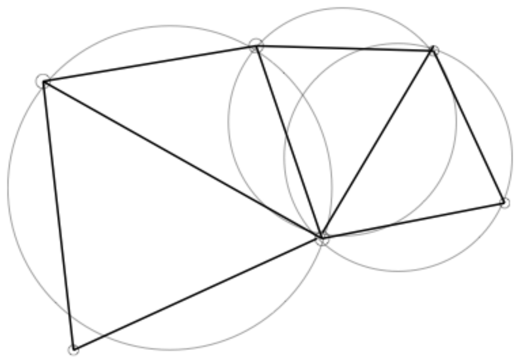
Delaunay Triangulation (1)

- Obs. Dual segments not necessarily intersect!



Delaunay Triangulation (2)

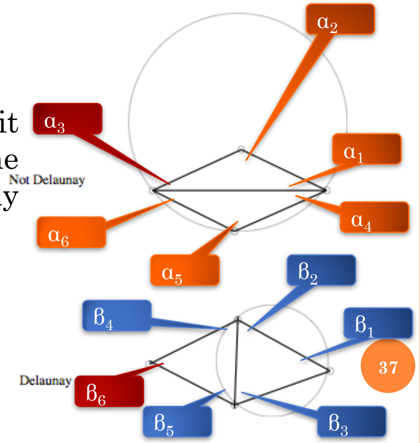
- Property: the circle circumscribed to a Delaunay triangle does not contain any site inside it



Delaunay Triangulation (3)

- Property: no segment can be illegal.
- A segment is *illegal* if:

  min α_i<min β_i
- If *e* is an illegal edge, then it is possible to swap the triangles to get a Delaunay triangulation.



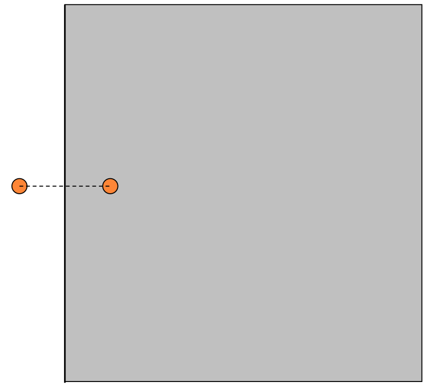
Delaunay Triangulation (4)

- Some papers exploit a Delaunay triangulation to route sensors towards a position contributing to a complete coverage.
- There are several algorithms to compute a Delaunay triangulation -> Possible lesson
- The Voronoi Diagram can be computed as dual construction of the Delaunay triangulation.
- Otherwise…

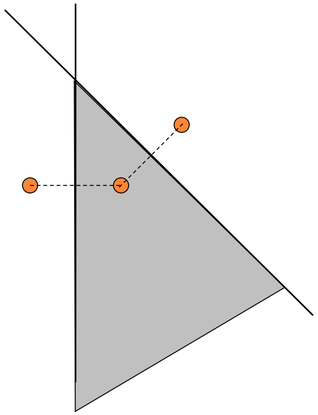
Algorithms to compute the Voronoi Diagram

Algorithm based on the Intersection of Half-planes (1)

A Voronoi cell can be obtained repeatedly intersecting opportune half-planes:

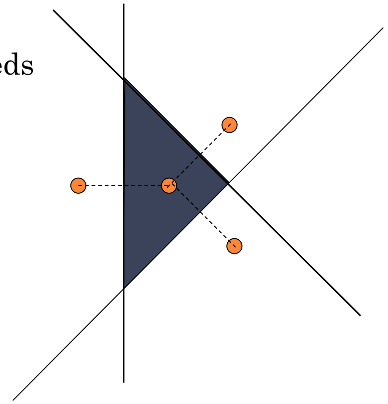


Algorithm based on the Intersection of Half-planes (2)



Algorithm based on the Intersection of Half-planes (3)

This operation needs to be iterated for each site.



Algorithm based on the Intersection of Half-planes (4)

How much does it cost to determine the intersection of a certain number k of halfplanes?
From computational geometry, a possible algorithm exploits the *divide-et-impera* technique…

Algorithm based on the Intersection of Half-planes (5)

Divide:
The set of k halfplanes is recursively split until k single halfplanes are obtained (Note: binary tree structure).
Impera:
The halfplane on each leaf is intersected with a rectangle R (the search space). In this way, each leaf contains now a polygon.
Combine:
Recursively, bottom-up, compute the intersection of two sibling polygons and put the result on the father node.

Algorithm based on the Intersection of Half-planes (6)

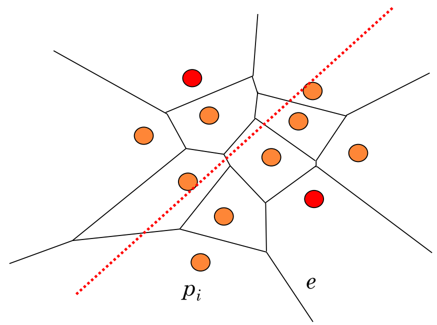
Time Complexity of *Combine*:
Let p and p' be the number of vertices of two general polygons that have to be intersected at some step of the algorithm. This can be done in $O(p+p')$ time.

It can be proved that the time complexity of the whole algorithm is $O(k \log k)$ – where k is the # of halfplanes to be intersected – and this is optimum because the sorting problem (using comparisons) can be reduced to the intersection of halfplanes.

Time Complexity of the whole algorithm to compute the Voronoi diagram: in order to find a single cell, $O(n)$ halfplanes need to be intersected, so $O(n \log n)$ per cell and $O(n^2 \log n)$ for the whole algorithm.

Intuition (1)

Not all the site pairs give raise to an axis!



Intuition (2)

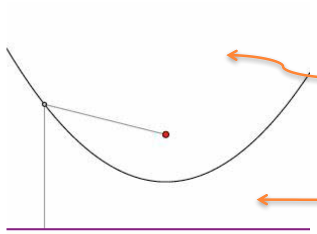
- Idea: use a well known technique in computational geometry.
- The sweep line is used to solve geometrical bidimensional problems through a sequence of almost onedimensional subproblems.
- Example: [Bentley, Ottmann'79] Compute the intersection points of n segments sweeping the plane with a horizontal line.
- When the sweep line moves, it encounters objects, and the algorithm solves the single problem related to each single object.

Intuition (3)

- This method cannot work as it is for Voronoi diagrams, because it would be necessary to "predict" the site position before the sweep line encounters them.
- Fortune [1986] designed an algorithm based on a different line, called beach line.

Fortune Algorithm (1)

- Idea: instead of considering the distance between sites, we introduce a line sweeping the plane (*sweep line*) helping us to compare distances.
- Somehow, this line "discovers" the Voronoi diagram on the just sweeped plane portion.
- Note. Given any point p and any external line l, the set of points equally distant from p and l is a parabola $P_{p,l}$.

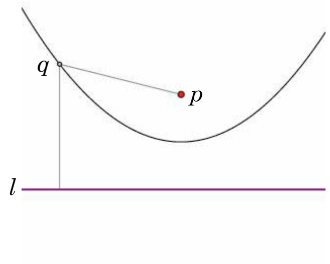


Points that are closer to p

Points that are closer to l

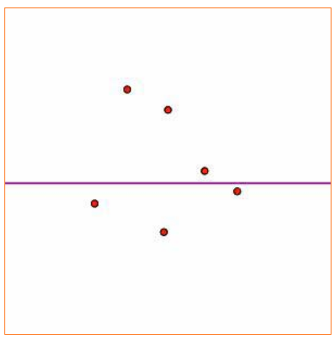
Fortune Algorithm (2)

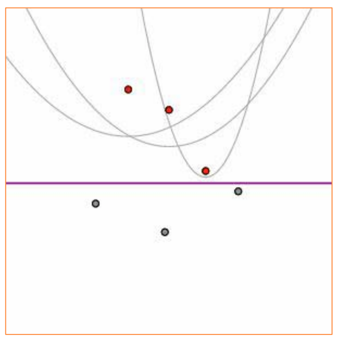
- Consider any point $q=(q_x, q_y)$.
- The sweep line is horizontal and its y-coordinate is l_y. Hence $dist(q,l)=l_y-q_y$.
- Given another point p, q lies on the parabola generated by p and l iff $dist(q,p)=l_y-q_y$.
- More in general:
  - $dist(q,p)< l_y-q_y$ if q lies above the parabola
  - $dist(q,p)=l_y-q_y$ if q lies on the parabola
  - $dist(q,p)>l_y-q_y$ if q lies under the parabola



Fortune Algorithm (3)

- The sweep line goes down.
- At each instant, consider the sites above the sweep line and the parabolas they define with the sweep line.
- Example:

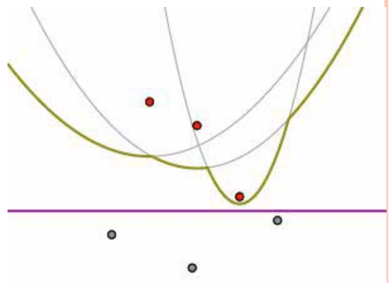




Fortune Algorithm (4)

- Define the *beach line* as the line formed by the union of the lower parabola arches.
- In other words: each vertical line crosses many parabolas; the lower intersection point belongs to the beach line.
- Note. Each arch of the beach line is associated with a site above the sweep line.

Nice videos at:
https://www.youtube.com/watch?v=7eCrHAv6sYY
https://www.youtube.com/watch?v=Y5X1TvN9TpM



To see an animation of the beach line:
http://www.raymondhill.net/voronoi/rhill-voronoi.html
(press the button "Animate sweep line")

FORTUNE ALGORITHM (5)

- Notice that if a point is above the beach line, it is closer to one of the sites above the sweep line than to the sweep line itself.
- In other words, this point lies inside the Voronoi cell of a site that the sweep line has already encountered.
- Hence, the Voronoi diagram above the beach line is completely determined by the sites above the sweep line.

FORTUNE ALGORITHM (6)

- Let us determine the condition such that the beach line passes through any point q.
- Let p_i be the i-th site and q be such that:

  $dist(q,p_1) \le dist(q, p_i)$ for any other i.
- Point q lies on the parabola generated by p_1 and l iff $dist(q, p_1) = l_y - q_y$.
- Joining the inequality and the condition:

  $dist(q, p_i) \ge dist(q,p_1) = l_y - q_y = dist(q,l)$.
- Remind that $dist(q,p) > dist(q,l)$ if q lies under $P_{p,l}$
- So q is on $P_{p1,l}$ and under any other parabola $P_{pi,l}$, so q is on the beach line. In other words:

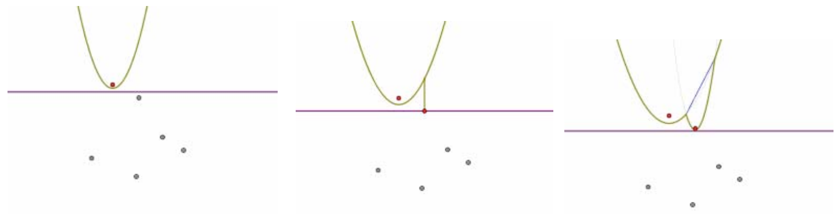
  When a point appears on the beach line, it is on the parabola associated to its closest site

FORTUNE ALGORITHM (7)

- Points on the beach line lying at the intersection of two parabola archs are called *breakpoints*.
- Breakpoints are at the same time closest to two sites. In other words,

  Breakpoints lie on the segments of the Voronoi diagram
- In order to construct the Voronoi diagram, it is enough to keep trace of the breakpoints.

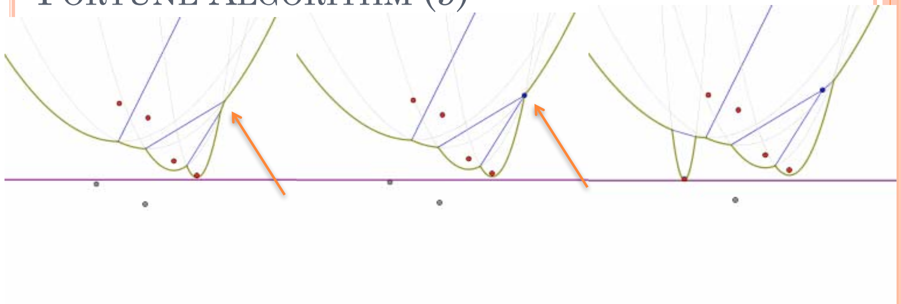
FORTUNE ALGORITHM (8)



Determine segments:

- A pair of breakpoints, corresponding to a segment in the Voronoi diagram, appear on the beach line exactly when the sweep line encounters a new site.
- We call this situation a *site event*.

Fortune Algorithm (9)

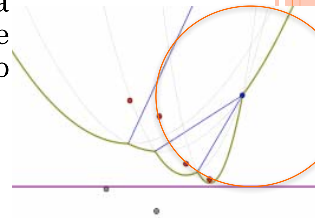


Determine vertices:

- While the sweep line moves, breakpoints move too, and they follow a segment; they reach a vertex when a parabola arch disappear.

Fortune Algorithm (10)

- It is easy to detect a new parabola arch appearing on the beach line: it appears when the sweep line encounters a site.
- Analogously, it is easy to detect a parabola arch disappearing from the beach line: when this arch is reduced to a single point x, it lies on 3 parabolas:
  - The one containing the disappearing arch
  - The one to its right
  - The one to its left



- So x is equally distant from 3 sites, corresponding to these 3 parabola arches -> a circle centered at x passes through these 3 sites.
- We determine a Voronoi vertex when the sweep line has finished to sweep this circle.
- We call this situation a *circle event.*

Fortune Algorithm (11)

Fortune Algorithm

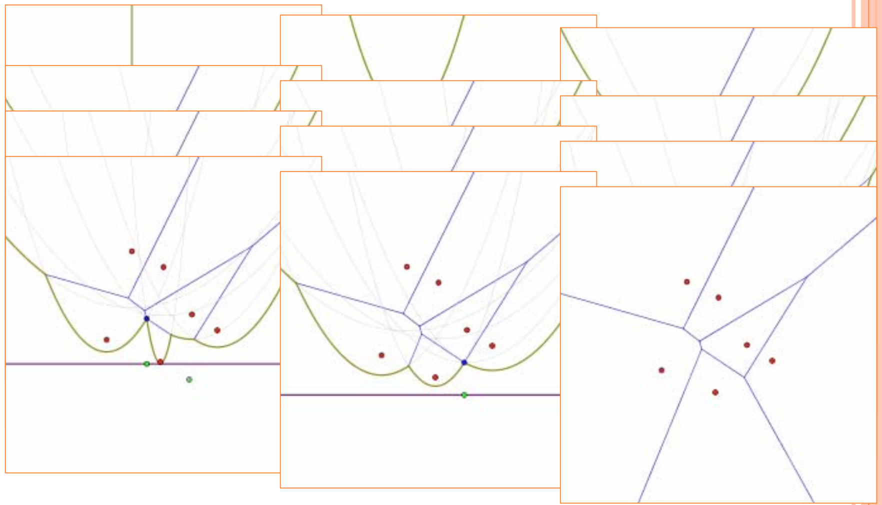
- Resume: In order to determine segments and vertices of the Voronoi diagram, we need to keep trace of the parabola arches appearing and disappearing on the beach line.
- We imagine to walk on the beach line left to right and we sort the order of the sites producing the parabola arches on it.
- This order cannot change until either a site event or a circle event happens.
- Breakpoints are implicitely stored as intersections of parabola arches on the beach line.

Fortune Algorithm (12)

Fortune algorithm (cntd.)

- If the next event encountered by the beach line is:
  - A site event, we insert the new site in the list of sites in the order indicated by its parabola arch and we store a new segment in the Voronoi diagram.
  - a circle event, we store both the new Voronoi vertex and the information that it is an extreme of the segments corresponding to two breakpoints joining in a single point.
- In both cases, we verify whether a new triple of sites producing a next circle event has been discovered.
- The Voronoi diagram is computed considering the (finite) sequence of these events.

FORTUNE ALGORITHM (13)



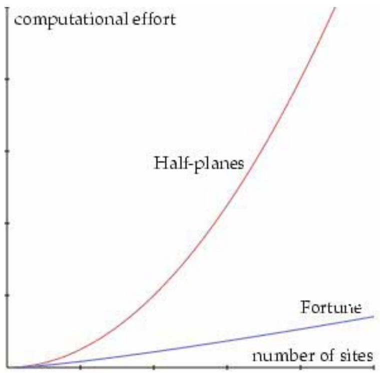
TIME COMPLEXITY

- The time complexity analysis of this algorithm follows tha typical scheme of all the algorithms based on the sweep line.
- Each event takes *O(1)* time to be detected + a constant number of accesses to the data structures to be stored.
- Each data structure contains *O(n)* information
- Each one of these accesses costs *O(logn)* time

- The whole time complexity is *O(n log n)*, and the occupied space is *O(n)*.
- This time complexity is optimum because the sorting problem (based on comparisons) can be reduced to the computation of the Voronoi diagram.

CONCLUSIONS

- Fortune algorithm is an efficient way to compute the Voronoi diagram.
- Whatever algorithm you use, it is reasonable to think that the time complexity grows up with the growth of the number of sites.
- The algorithm based on the intersection of halfplanes runs in $O(n^2 \log n)$ time if there are n sites.
- Fortune algorithm runs in $O(n \log n)$ time.



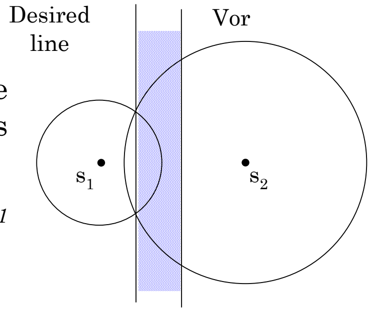
HETEROGENEOUS SENSORS

Heterogeneous Sensors

- Sensors are not necessarily all equal. We speak about a heterogeneous sensor network if:
  - the devices are different
  - The sensing and communicating ability of the sensors depend on their position (not smooth terrain, obstacles, …)
- The previously described approaches (based on virtual forces and on Voronoi cells) do not work well with heterogeneous sensors:
  - Virtual forces: forces depend on the distance
  - Voronoi: cells do not take into account the coverage capability

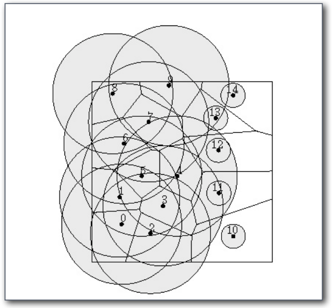
Limitations of the Protocols based on Voronoi Cells (1)

- The protocol based on the construction of Voronoi cells would assign:
  - The left halfplane to s_1 (included the blue zone)
  - The right halfplane to s_2

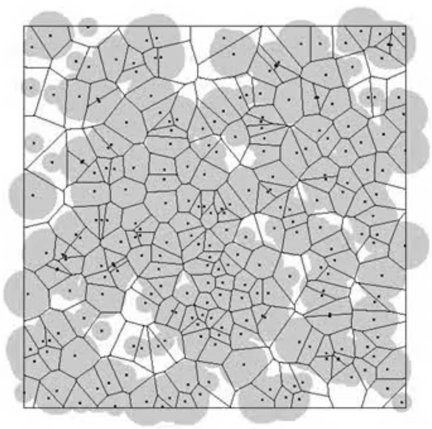


Limitations of the Protocols based on Voronoi Cells (2)

- Stale situation:
  - the sensors on the left (big circles) do not move since they completely cover their cells
  - the sensors on the right (small circles) do not move since their circles are completely used to cover a portion of their cell (in other words, their coverage capacity is maximized).

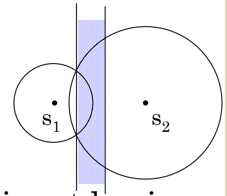


Limitations of the Protocols based on Voronoi Cells (3)



A NEW NOTION OF DISTANCE

- In the known algorithms, the heterogeneity is ignored
- We introduce a new notion of distance keeping into account:
  - The Euclidean distance
  - The heterogeneity of the devices



- There are many possibilities, but we aim at having:
  - Diagrams with straigh edges (convex polygons)
  - a distance whose set of points equally distant from two sensors contains the intersection of their sensing circles

LAGUERRE DISTANCE (1)

W. Blaschke. Vorlesungen uber Differentialgeometrie III. Springer Berlin. 1929.

- Defined in $\mathscr{R}^3$
- Given two points $P=(x,y,z)$ and $Q=(x',y',z')$, their Laguerre distance is:
  - $d_L^2(P,Q)=(x-x')^2+(y-y')^2-(z-z')^2$
- P can be seen as the (oriented) circle centered at (x,y) and having radius $|z|$

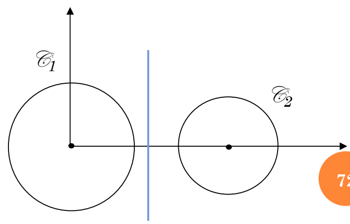
LAGUERRE DISTANCE (2)

- Given two circles $\mathscr{C}_1$ and $\mathscr{C}_2$, centered at C_1 and C_2 respectively, and with radii r_1 and r_2, their Laguerre distance is:
  - $d_L^2(\mathscr{C}_1, \mathscr{C}_2)= d_E^2(C_1, C_2) - (r_1 - r_2)^2$
- The Laguerre distance between a point $P=(x,y)$ and a circle $\mathscr{C}=(x',y',r)$ is:
  - $d_L^2(P,C)=(x-x')^2+(y-y')^2-r^2$

LAGUERRE DISTANCE (3)

- Lemma. Given two circles $\mathscr{C}_1$ and $\mathscr{C}_2$ centered at C_1 and C_2 ($C_1 \neq C_2$) and radii r_1 and r_2, the sets of point equally distant from $\mathscr{C}_1$ and $\mathscr{C}_2$ is a straight line (called radical axis) orthogonal to the segment joining C_1 and C_2 and at distance k from C_1, where

$$k = \frac{d_E(C_1,C_2)}{2} + \frac{r_1^2 - r_2^2}{2d_E(C_1,C_2)}$$



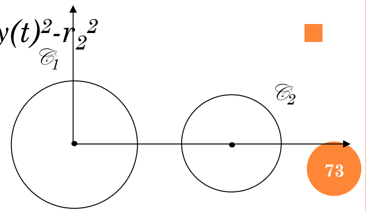
LAGUERRE DISTANCE (4)

Proof. Consider the set of points $P(t)=(x(t),\ y(t))$ equally distant from $\mathscr{C}_1$ and $\mathscr{C}_2$, i.e. such that

$$d_L(P(t),\ \mathscr{C}_1)=d_L(P(t),\ \mathscr{C}_2).$$

- If $C_1=C_2$ and $r_1=r_2 \Rightarrow P(t)$ is the whole plane
- If $C_1=C_2$ and $r_1 \neq r_2 \Rightarrow P(t)$ is the empty set
- If $C_1 \neq C_2$:

$x(t)^2+y(t)^2-r_1^2=(d_E(C_1,C_2)-x(t))^2+y(t)^2-r_2^2$



$d_L^2(P,C)=(x-x')^2+(y-y')^2-r^2$

LAGUERRE DISTANCE (5)

- Lemma. Given two circles $\mathscr{C}_1$ and $\mathscr{C}_2$ centered at C_1 and C_2 ($C_1 \neq C_2$) and having radii r_1 and r_2, theri centers lie on the same side w.r.t. the radical axis if and only if

  $d_E^2(C_1,C_2) < |r_1^2-r_2^2|.$

Proof. The axis can lie either to the right or to the left.

- Right:

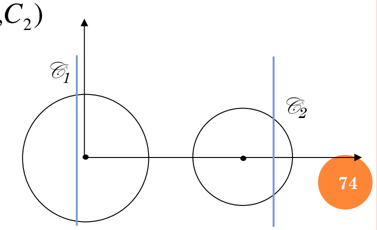
$$k=\frac{d_E(C_1,C_2)}{2}+\frac{r_1^2-r_2^2}{2d_E(C_1,C_2)} \geq d_E(C_1,C_2)$$

$d_E^2(C_1,C_2) \leq r_1^2-r_2^2 \Rightarrow r_1 \geq r_2$

- Left:

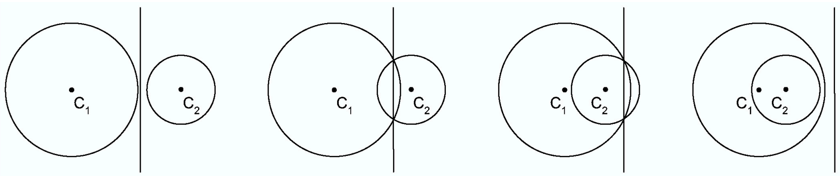
$$\frac{d_E(C_1,C_2)}{2}+\frac{r_1^2-r_2^2}{d_E(C_1,C_2)} \leq 0$$

$d_E^2(C_1,C_2) \leq r_2^2-r_1^2 \Rightarrow r_2 \geq r_1$



LAGUERRE DISTANCE (6)

- Possible positions of the radical axis of two cricles $\mathscr{C}_1$ and $\mathscr{C}_2$

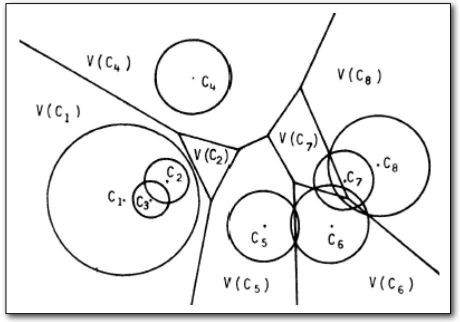


VORONOI-LAGUERRE DIAGRAM (1)

Voronoi-Laguerre diagram of $\mathscr{C}_1, \ldots, \mathscr{C}_n$:

- $V_i = \cap \{p \in \mathscr{R}^2 \mid d_L^2(\mathscr{C}_i,P) \leq d_L^2(\mathscr{C}_j,P)\}$

H. Imai, M. Iri, K. Murota. "Voronoi Diagram in the Laguerre Geometry and its Applications". SIAM J. Comput. 14(1), 93-105. 1985.



They have similarities and differences w.r.t. the classical Voronoi diagrams…

VORONOI-LAGUERRE DIAGRAM (2)

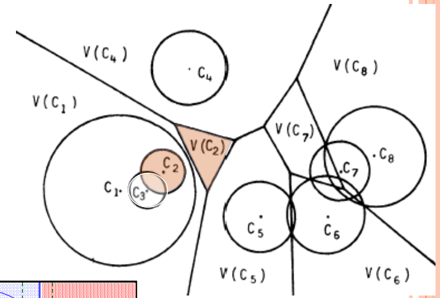
Similarities:

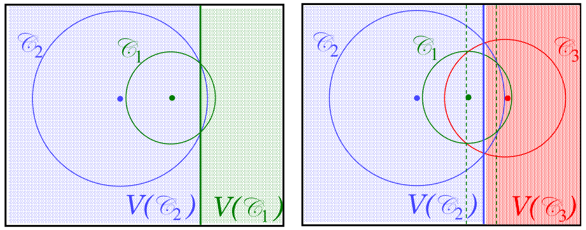
- Voronoi-Laguerre polygons partition the plane
- V_i is always convex because it is the intersection of some halfplanes
- if $r_i=0$ for each $i=1, \ldots, n$, theVoronoi-Laguerre diagram is in fact the classical Voronoi diagram.

VORONOI-LAGUERRE DIAGRAM (3)

Differences:

- $\mathscr{C}_i$ can be external to V_i (see $\mathscr{C}_2$)
- V_i can be empty (e.g. if $\mathscr{C}_i$ is inside the union of other circles - see $\mathscr{C}_3$)





VORONOI-LAGUERRE DIAGRAM (4)

- Theorem. Given n circles $\mathscr{C}_i$ centered at $C_i=(x_i,y_i)$ and having radii r_i, $i=1, \ldots, n$, let V_i be their Voronoi-Laguerre polygons.

  For each i and j, $V_i \cap \mathscr{C}_j \subseteq \mathscr{C}_i$.

  In other words, the intersection of V_i with a circle $\mathscr{C}_j$ is included in $\mathscr{C}_i$.

VORONOI-LAGUERRE DIAGRAM (5)

Proof. By contradiction, assume that ther exists a point $P \subseteq V_i$ in $\mathscr{C}_j$ but non in $\mathscr{C}_i$, for some $j \neq i$.

- Since $P \subseteq V_i$ it holds $d_L(P, \mathscr{C}_i) < d_L(P, \mathscr{C}_j)$ for each $j \neq i$, i.e.

$$d_E^2(P, \mathscr{C}_i)-r_i^2<d_E^2(c_j,P)-r_j^2$$

- Since P is in $\mathscr{C}_j$ but non in $\mathscr{C}_i$,

$$d_E^2(P, \mathscr{C}_j)\leq r_j^2 \text{ and } d_E^2(P, \mathscr{C}_i) \geq r_i^2$$

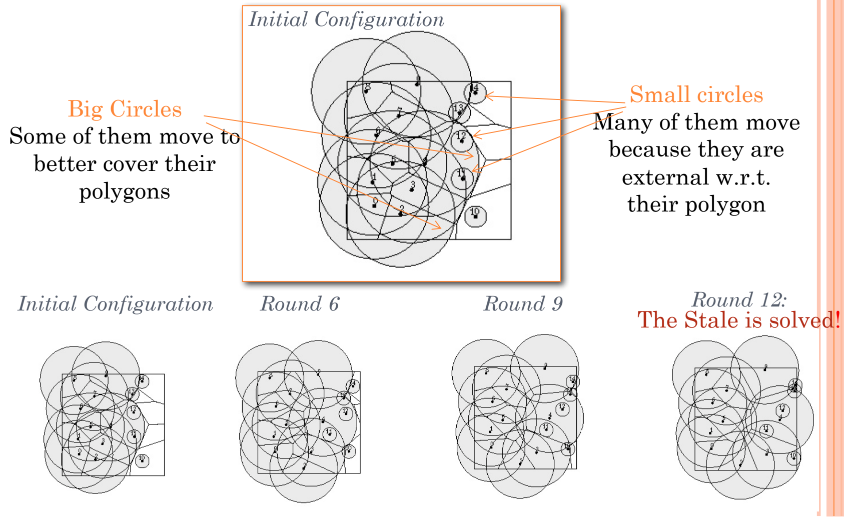
- Combining: 0<0 Contradiction. ■

ALGORITHM BASED ON VORONOI-LAGUERRE DIAGRAM (1)

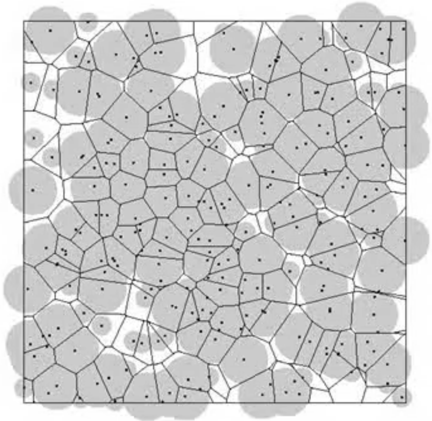
Algorithm executed by each sensor s_i:

- Compute V_i
- If s_i is inside V_i, move toward the minimax (by at most by $d_i^{max}=r_{tx}/2\text{-}r_i$ where $r_{tx}=min_i\, r_i^{tx}$) if the coverage of V_i is increased
- If s_i is outside V_i, move toward the minimax (by at most $d_i^{max}=r_{tx}/2\text{-}r_i$)
- if V_i is empty, do nothing.

ALGORITHM BASED ON VORONOI-LAGUERRE DIAGRAM (2)

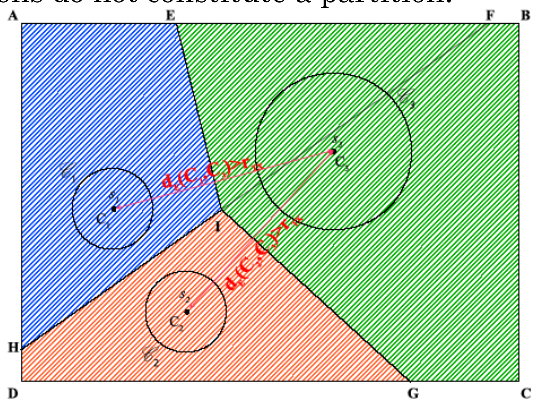


ALGORITHM BASED ON VORONOI-LAGUERRE DIAGRAM (3)



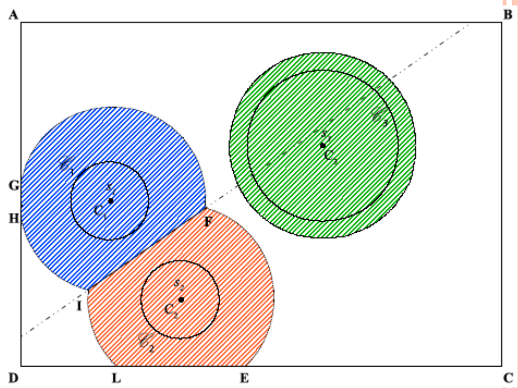
PROPERTIES OF THE ALGORITHM (1)

- Obs.:
  - "local" polygon≠"global" polygon and the set of local polygons do not constitute a partition!



PROPERTIES OF THE ALGORITHM (2)

- We define a curve polygon V'_i generated intersecting the "local" polygon with the circle of radius $d_i^{max}+r_i=r_{tx}/2$.



PROPERTIES OF THE ALGORITHM (3)

- Lemma. $V'_i \cap V'_j = \phi \ \forall i \neq j$
- Lemma. $\forall i \neq j,\ V'_i \cap \mathcal{C}_j \subseteq \mathcal{C}_i$.

  In other words, each curve polygon can be covered by the sensor generating it better than by any other sensor.

PROPERTIES OF THE ALGORITHM (4)

Th. The algorithm converges.

Proof. Let $V'_i{}^{(k)}$ be the curve polygon of s_i at round k.

- Let $A_i^{(k)}$ and $A_i^{(k)}(s_i)$ be the areas covered inside $V'_i{}^{(k)}$ by all the sensors and by the sole sensor s_i at round k, respectively. Let $A_i'^{(k)}$ be the covered area considering the positions of the sensors at round $k+1$.
- Obs. $A_i'^{(k)} \neq A_i^{(k+1)}$
- Let $A^{(k)}_{total}$ be the area covered by the AoI by all the sensors.
- We have to prove that $A^{(k)}_{total} < A^{(k+1)}_{total}$

PROPERTIES OF THE ALGORITHM (5)

Proof. (cntd.)

- $\mathscr{P}^{(k)}=\{V'_1{}^{(k)}, V'_2{}^{(k)}, \ldots, AoI \setminus \cup_i V'_i{}^{(k)}\}$ is a partition of the AoI.
- $AoI \setminus \cup_i V'_i{}^{(k)}$ is constituted by points that are uncovered and cannot be covered in a single round; it does not contribute to $A^{(k)}_{total}$.
- $A^{(k)}_{total} = \Sigma_i A_i^{(k)}$
- $A_i^{(k)} = A_i^{(k)}(s_i)$ (by the previous lemma)
- $A_i^{(k)}(s_i) < A_i'^{(k)}(s_i)$ (by the algorithm)
- $A_i'^{(k)}(s_i) \leq A_i'^{(k)}$
- Hence: $A^{(k)}_{total} = \Sigma_i A_i^{(k)} < \Sigma_i A_i'^{(k)}$
- Since the coverage at round $k+1$ does not depends on the partition:

  $\Sigma_i A_i'^{(k)} = A^{(k+1)}_{total}$

Properties of the Algorithm (6)

- Convergence does not imply termination.
- In order to guarantee termination, we introduce a minimum movement threshold ε, so that sensors do not move if they are suppose to do by less than ε.
- Corollary. The algorithm, with the addition of the minimum movement threshold, terminates.

Open Problems

- Obstacles and terrain asperities
  - Anisotropy
  - Movement obstacles
- AOI with complex shape
  - concave regions and corridors
- …