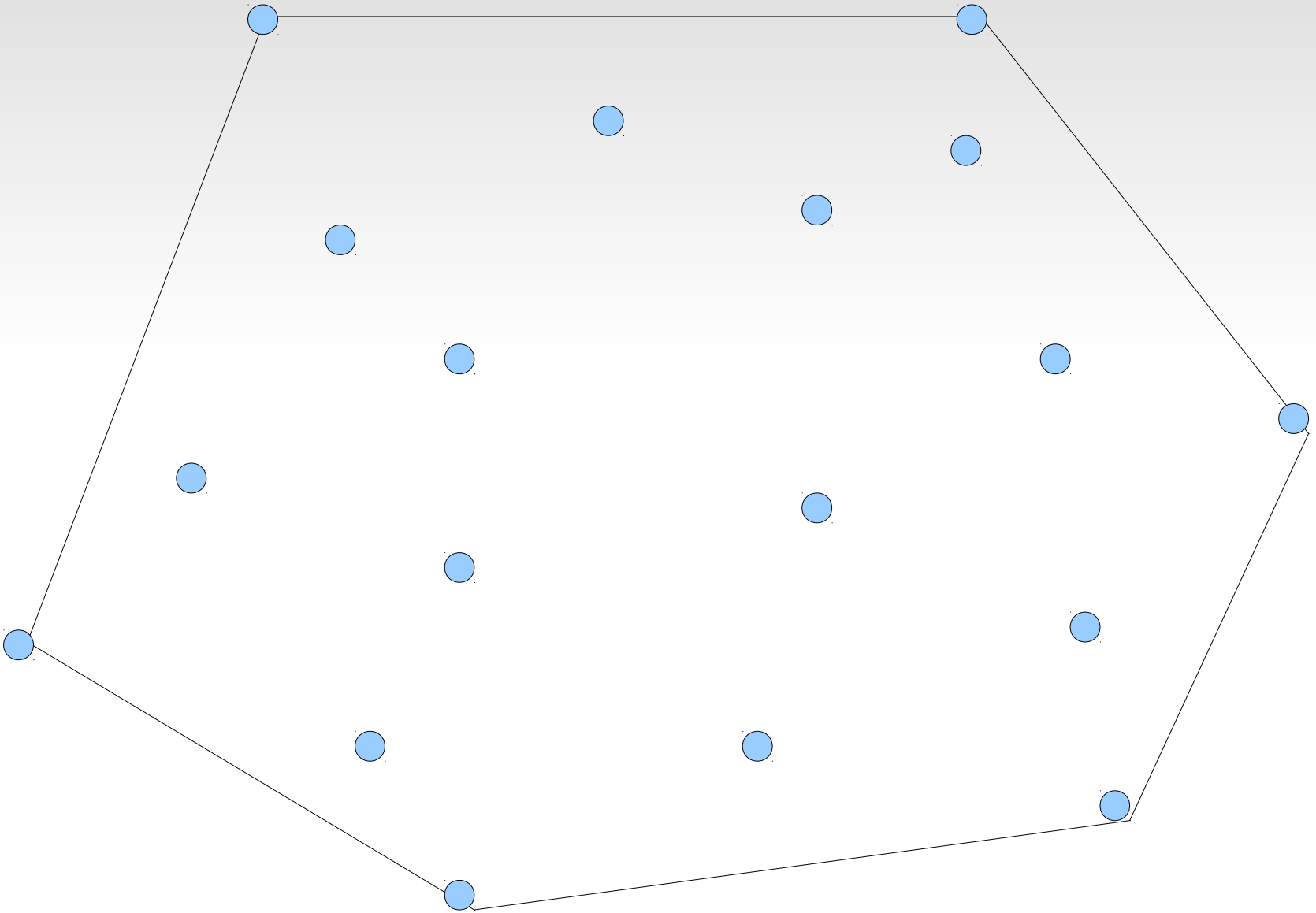
Enveloppes Convexes

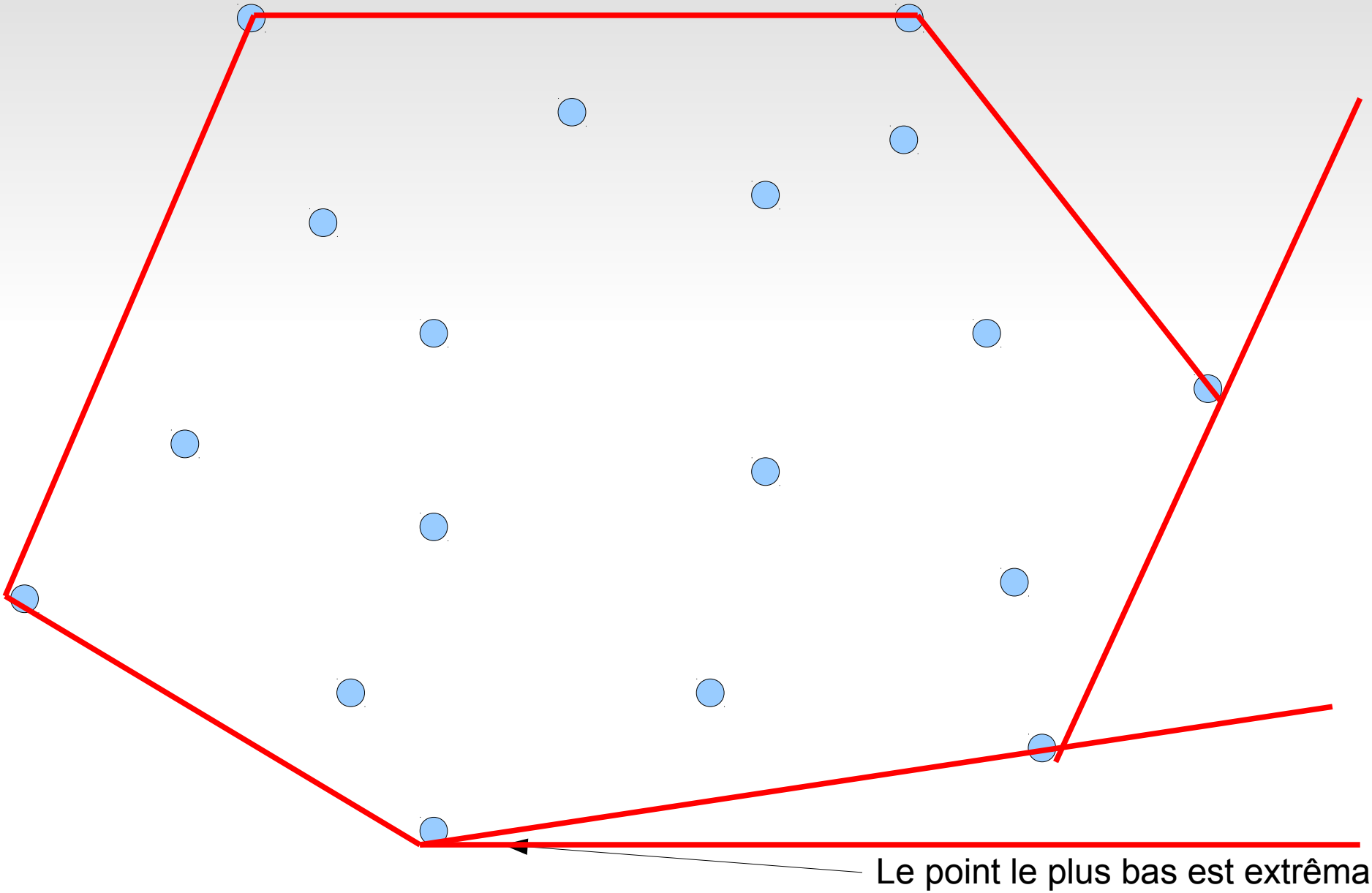
Voronoï - Delaunay

Marc Neveu

Enveloppe Convexe



Jarvis : Point Extremal+ Tangente

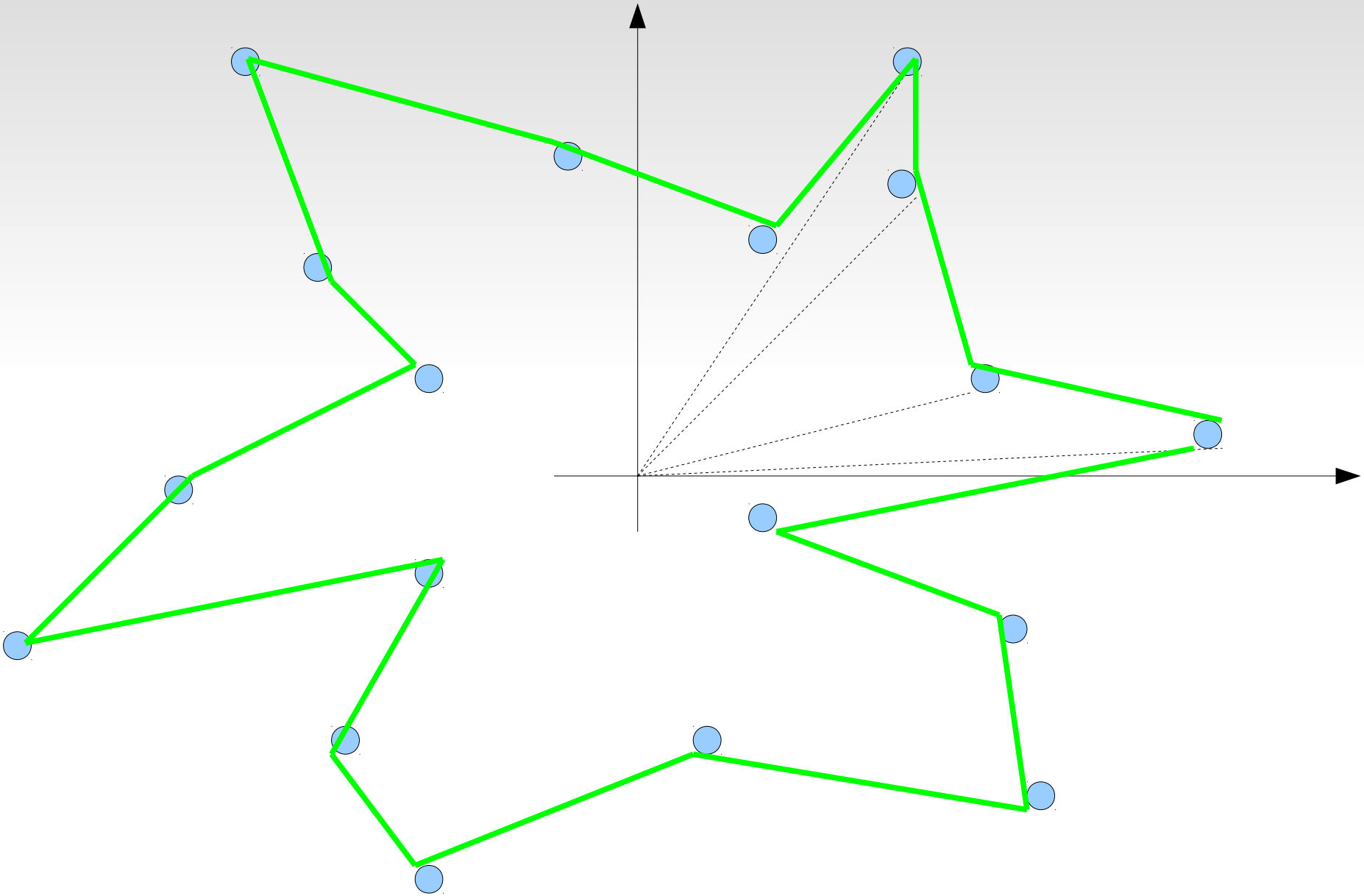


Algorithme de Jarvis

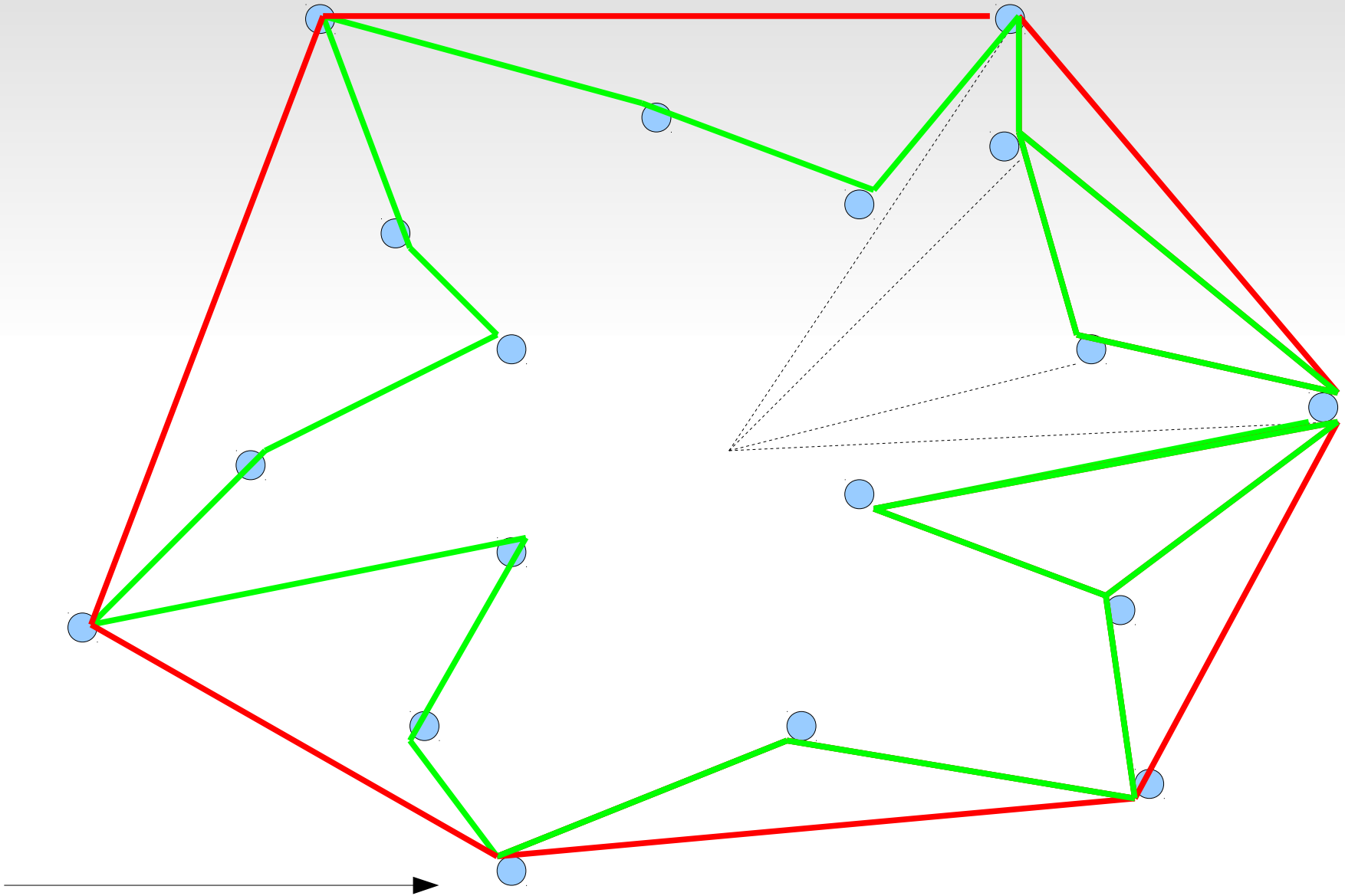
```
entrée : S un ensemble de points.
u = le point le plus bas de S;
min = ∞
Pour tout w ∈ S \ {u}                                          O(n)
    si angle(ux, uw) < min alors {min = angle(ux, uw); v = w;}
u.suivant = v;
Faire
    S = S \ {v}
    Pour tout w ∈ S
        min = ∞                                                O(n²)
        si angle(v.pred v, vw) < min alors
            {min = angle(v.pred v, vw); v.suivant = w;}
        v = v.suivant;
Tant que v <> u
```

Complexité : $O(n^2)$

Graham : Pt intérieur + Tri



Graham : +parcours depuis le plus bas



Algorithme de Graham

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; — O(1)

trier S autour de l'origine; — O(n logn)

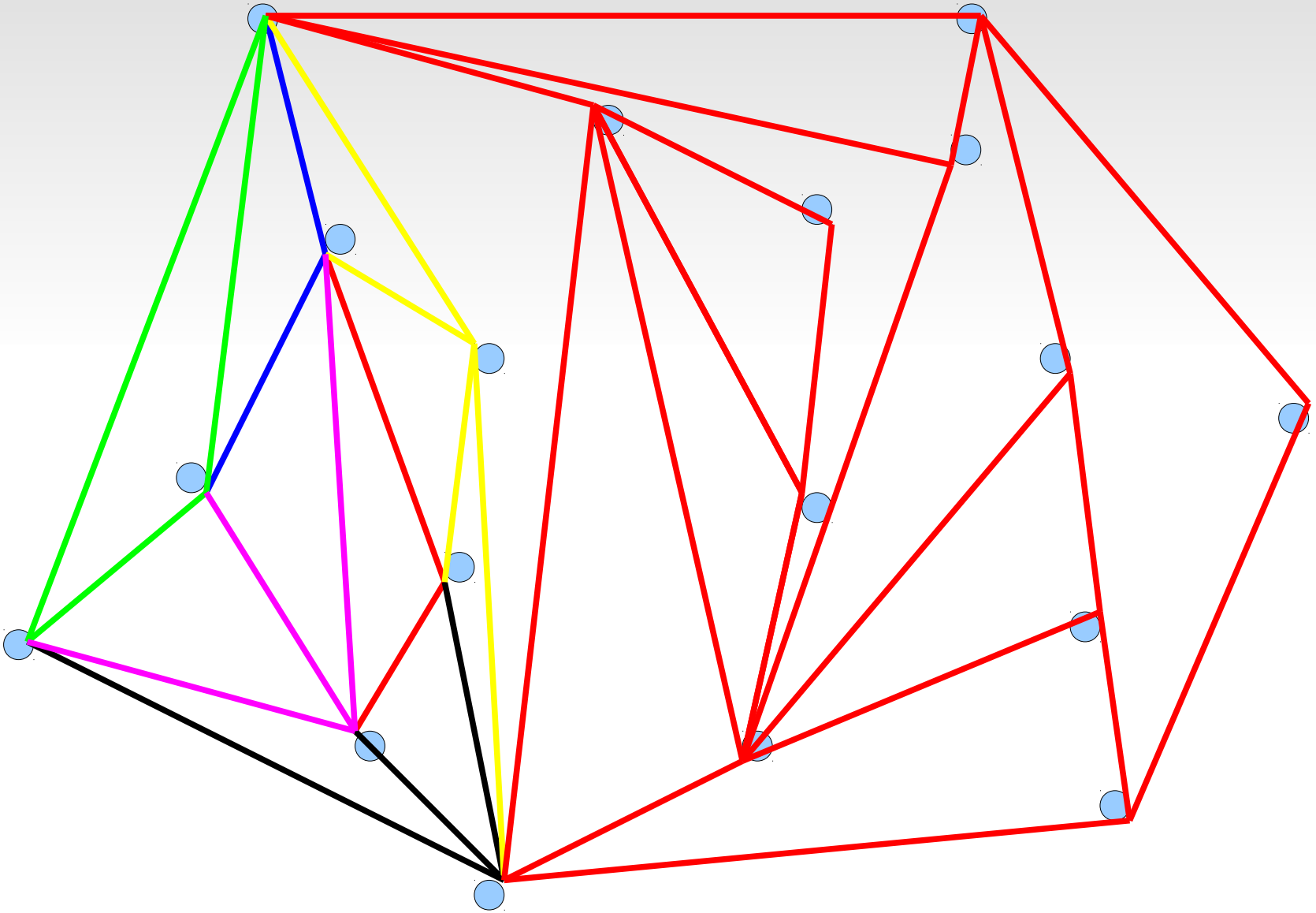
u = le point le plus bas de S; — O(n)

```
v = u;
tant que v.suivant <> u
    si (v, v.suivant, v.suivant.suivant)= touràgauche
        v = v.suivant;
    sinon
        v.suivant = v.suivant.suivant;
        si v <> u     v = v.precedent;
```

O(n)

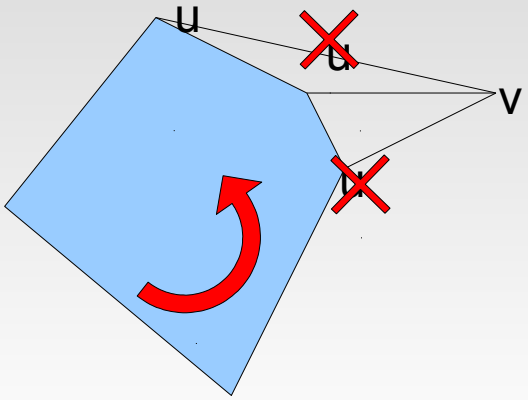
Complexité : O(n log n)

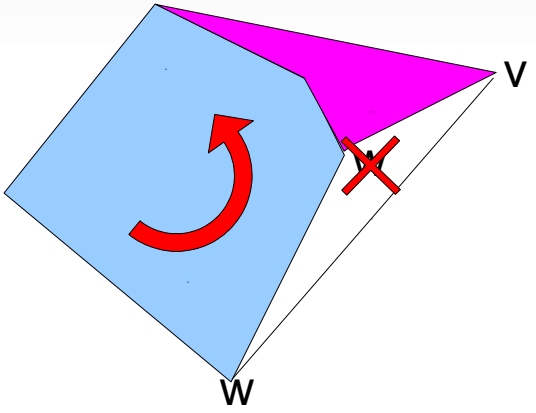
Enveloppes : tri en x

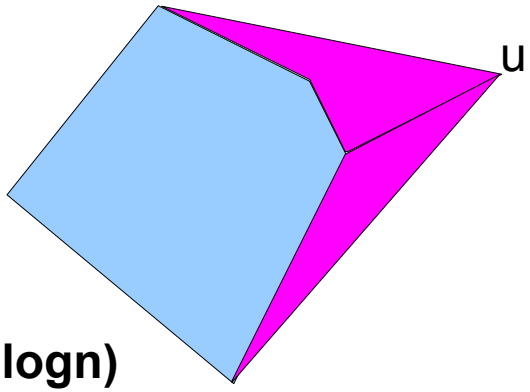


Algorithme par tri en x

entrée : S un ensemble de points.
trier S en x;
initier une liste circulaire avec les 3 points les plus à gauche tel que u àdroite et u, u.suivant, u.suivant.suivant tourneàgauche;
Pour v le prochain en x
    w=u
    tant que (v, u, u.suivant) tourneàdroite
        u = u.suivant;
    v.suivant = u; u.pred = v;
    tant que (v, w, w.pred) tourneàgauche
        w = w.pred;
    v.pred = w; w.suivant = v;
    u = v;



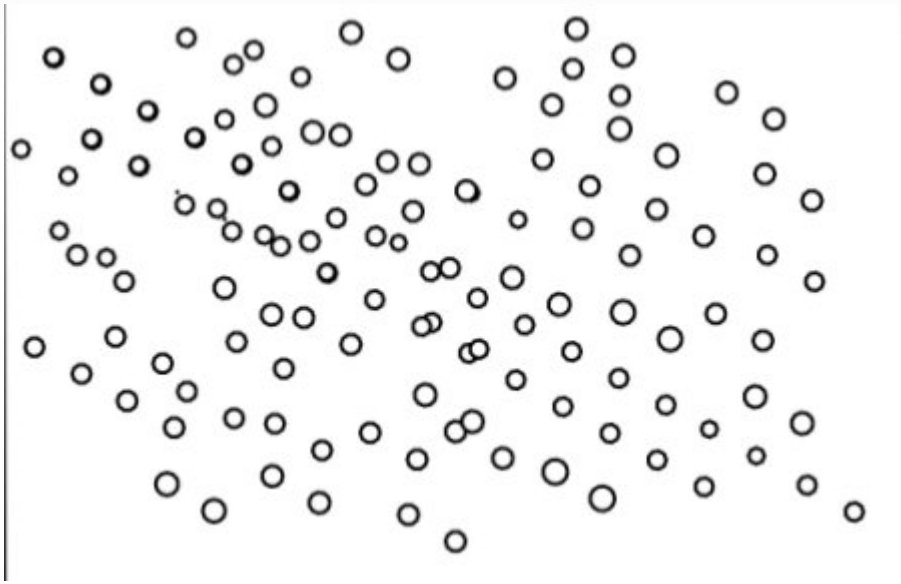




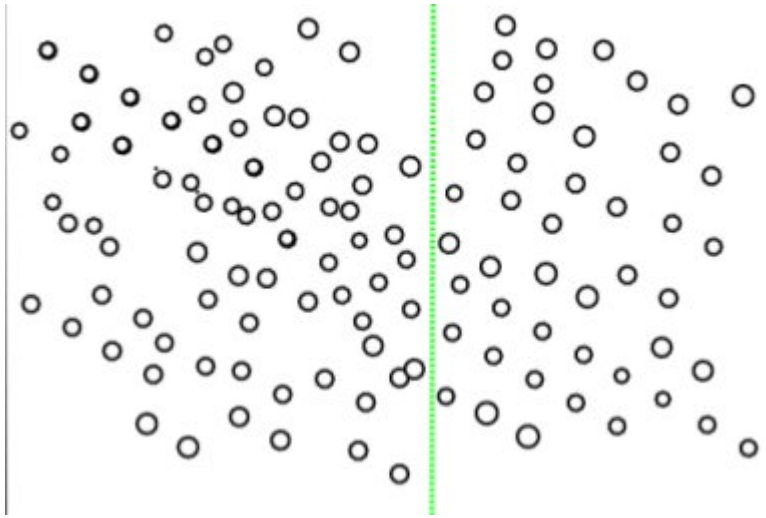
O(nlogn) Arete dans la triangulation ~3n **Complexité : O(nlogn)**

Diviser pour Régner

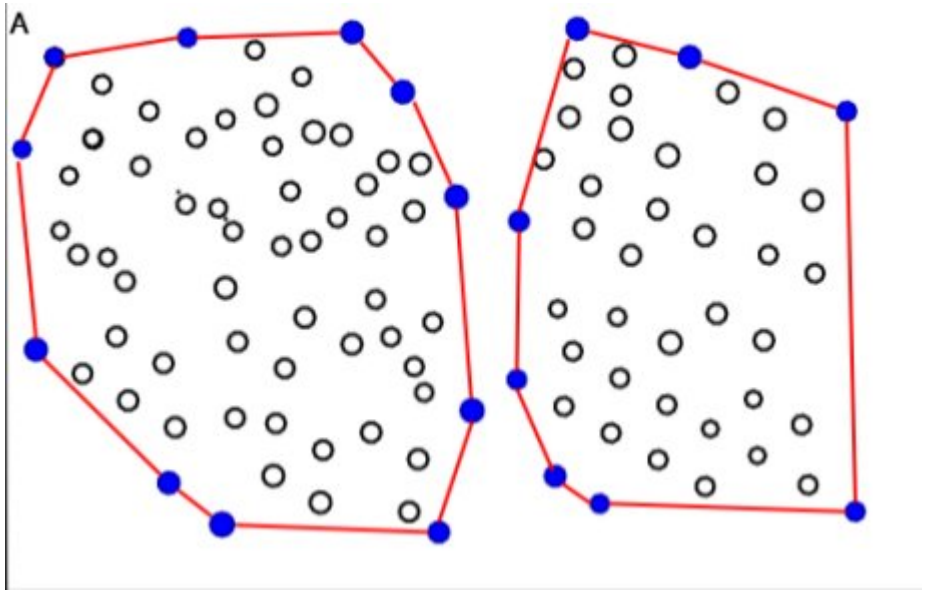
- Données : nuage de points M(x,y)



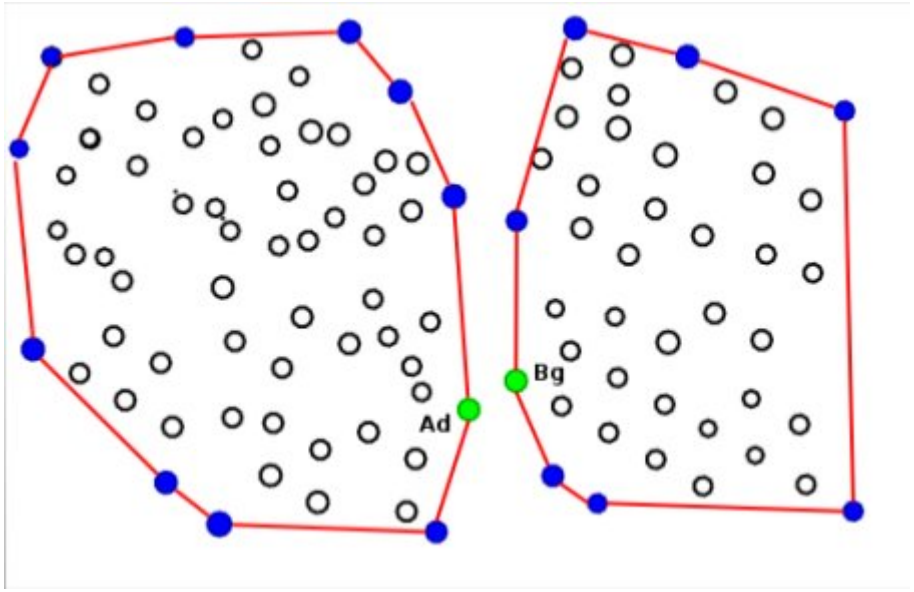
- Trier d'abord les points M = (x , y) suivant leurs abscisses, et les ranger en une liste p1 , . . . , pn
- On brise la liste en deux sous-listes de même longueur p1 , . . . , pk et p+1 , . . . , pn



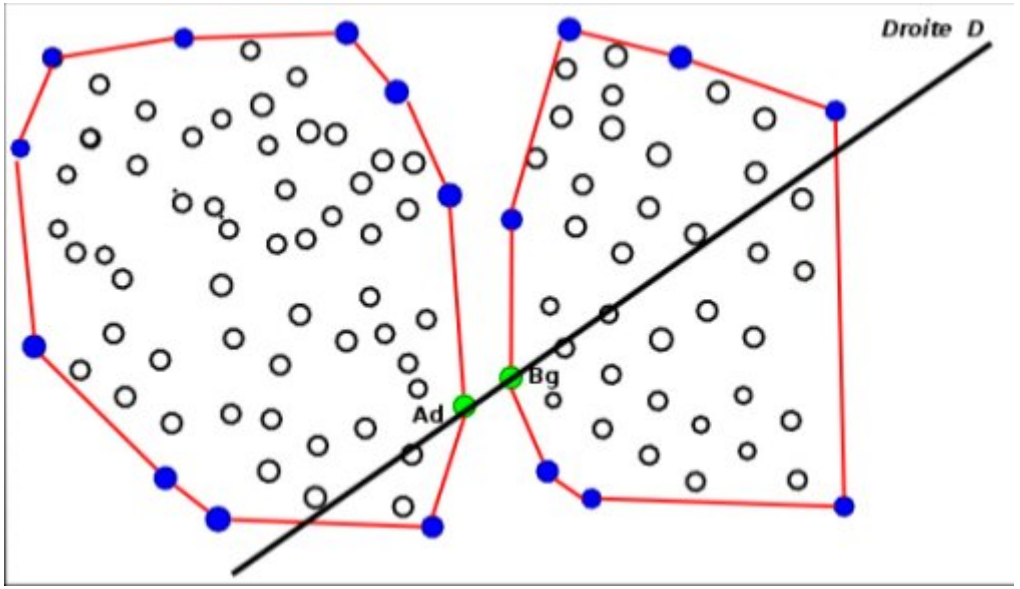
- On fabrique récursivement les enveloppes convexes des deux sous-listes A et B de E . on s'arrête si une sous-liste de A ou B comporte moins de trois points.



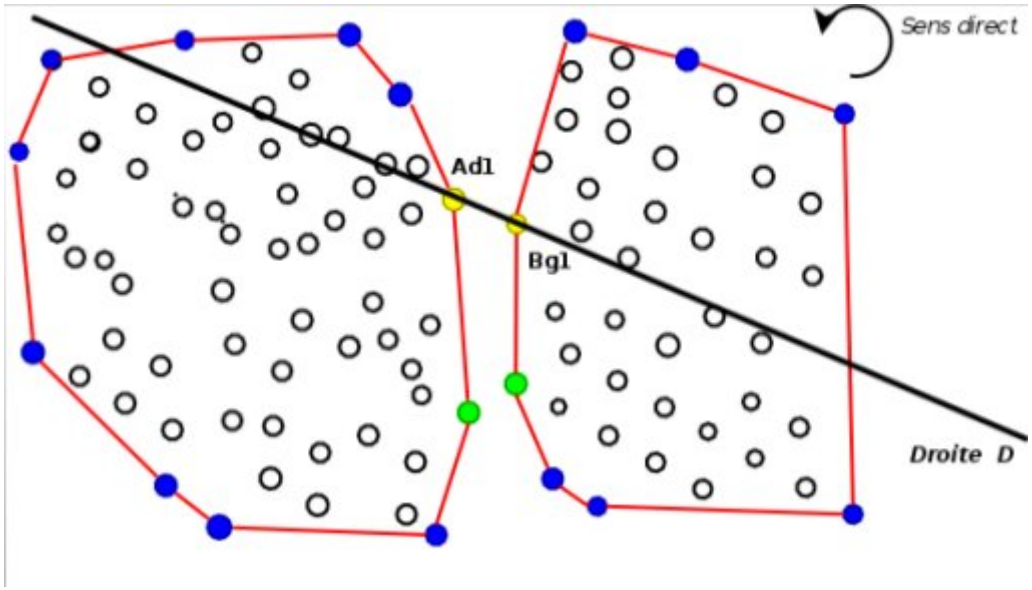
- Pk est le point max à droite de l'enveloppe convexe de gauche A. Pk+1 est celui le plus à gauche de l'enveloppe de droite B .
- On introduit deux variables Ad et Bg , initialisées respectivement aux valeurs pk et pk +1 .



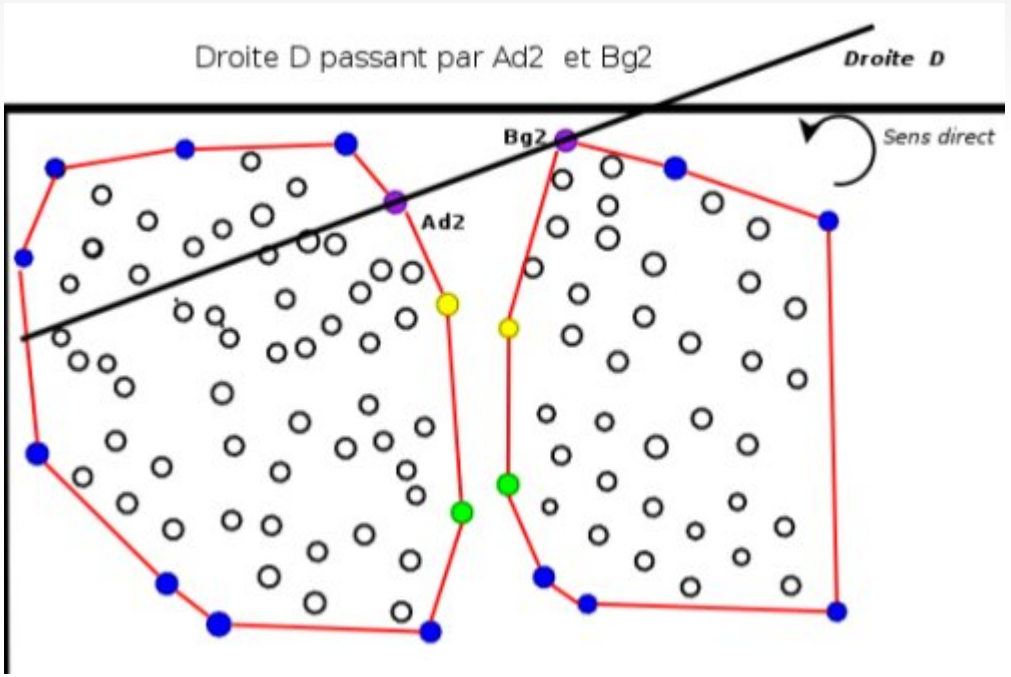
- Si le point qui suit Ad en parcourant A dans le sens direct est au dessus de D, la droite passant par Ad et Bg . On remplace Ad par son successeur noté Ad1.



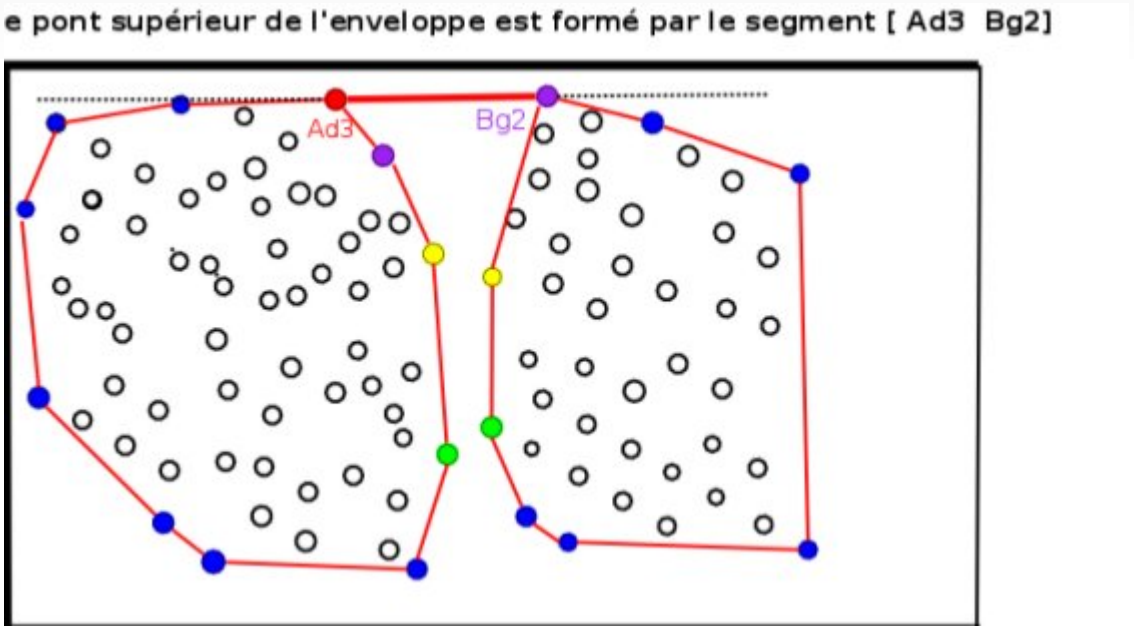
- Si le point qui précède Bg sur B parcourue dans le sens direct est au dessus de D . On remplace Bg par son prédecesseur noté Bg1.
-



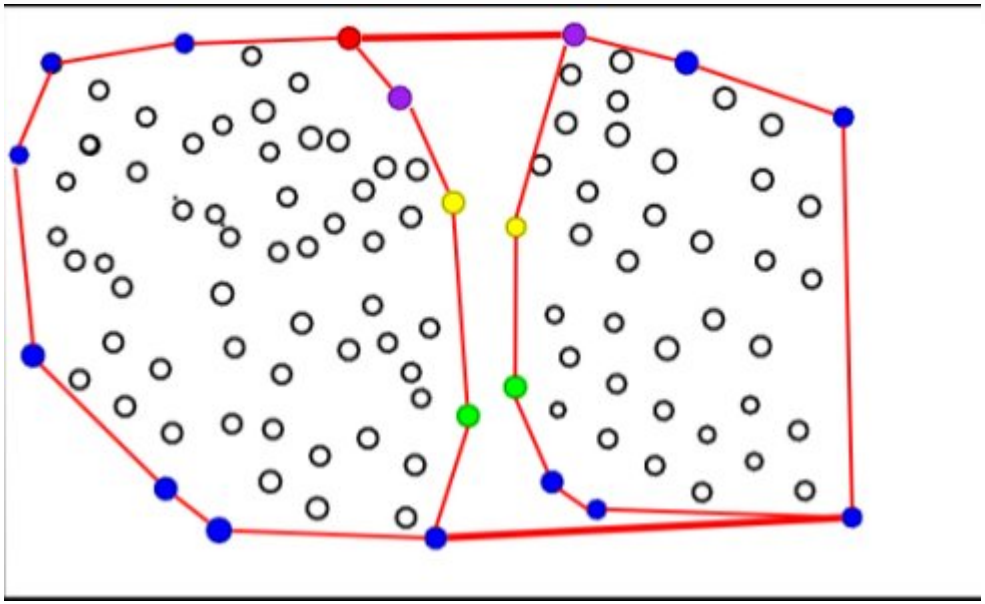
- Itération des deux actions autant que possible
-



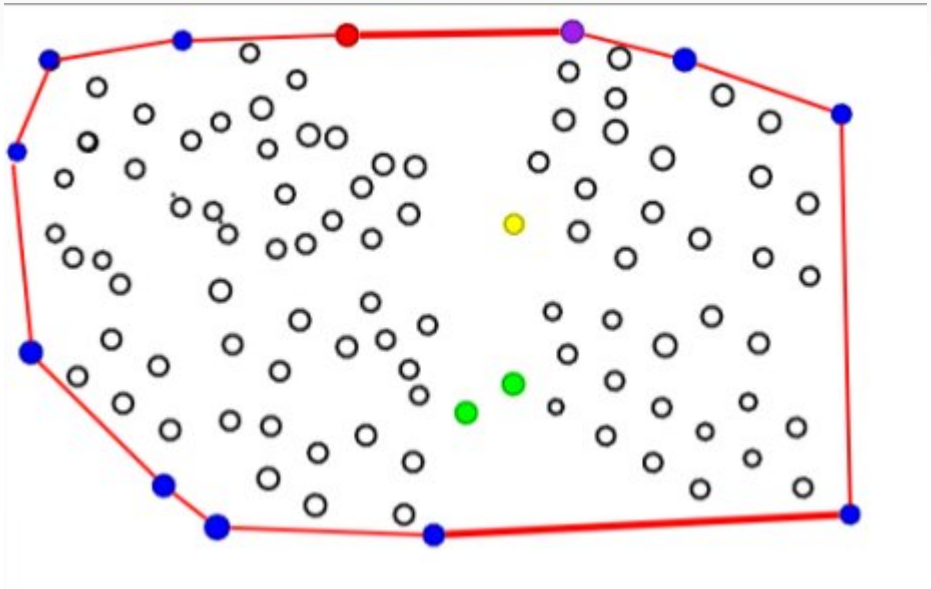
- la droite D fournit enfin le pont supérieur
-



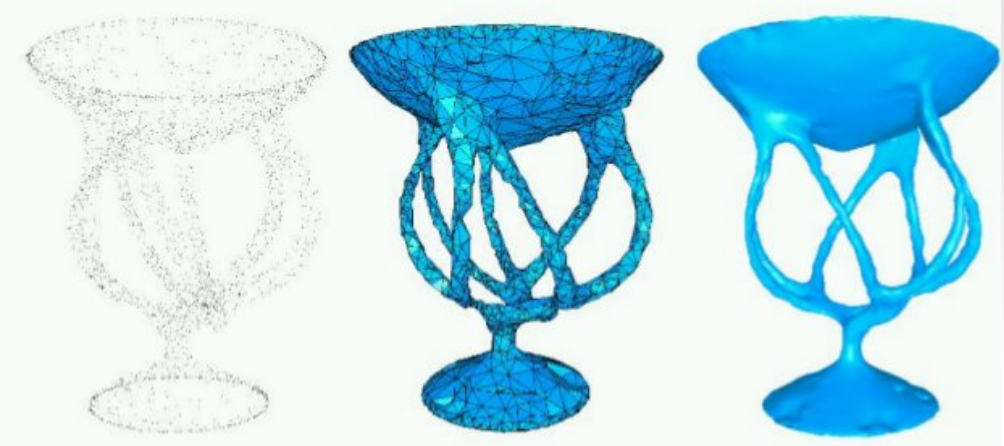
- la droite D fournira de même le pont inférieur
-



- On a enfin notre enveloppe convexe.
-

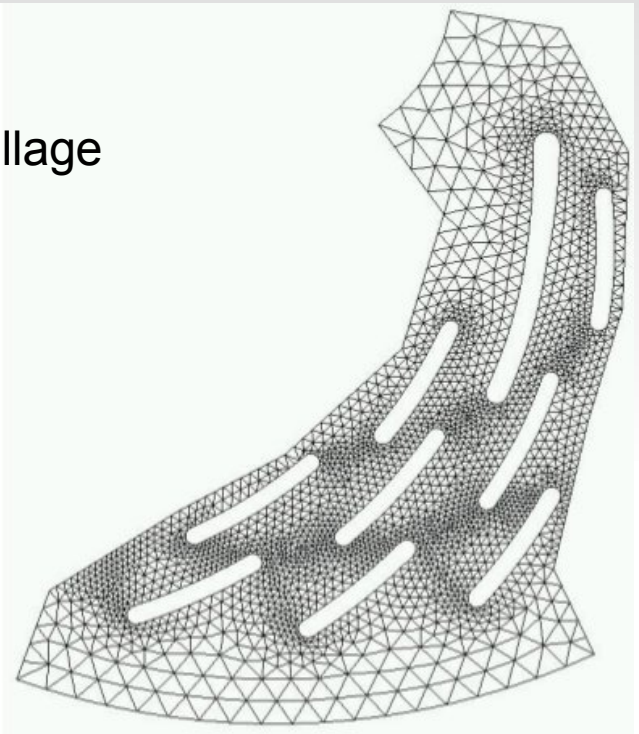


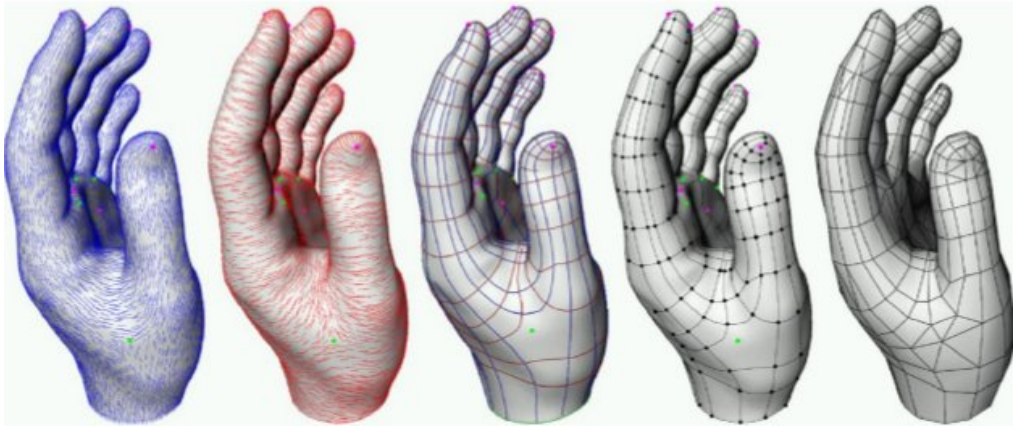
Voronoï - Delaunay



Reconstruction

Maillage

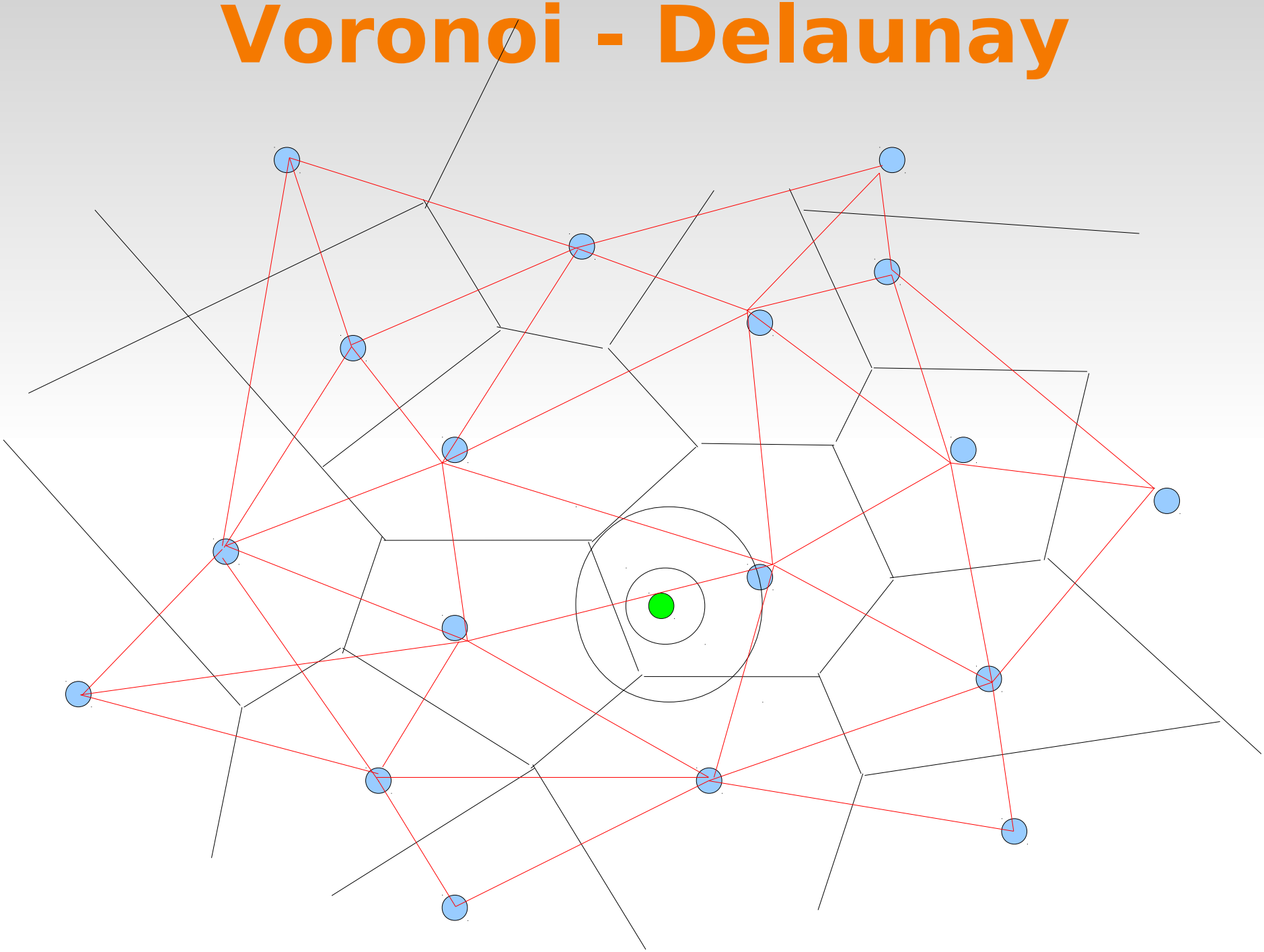




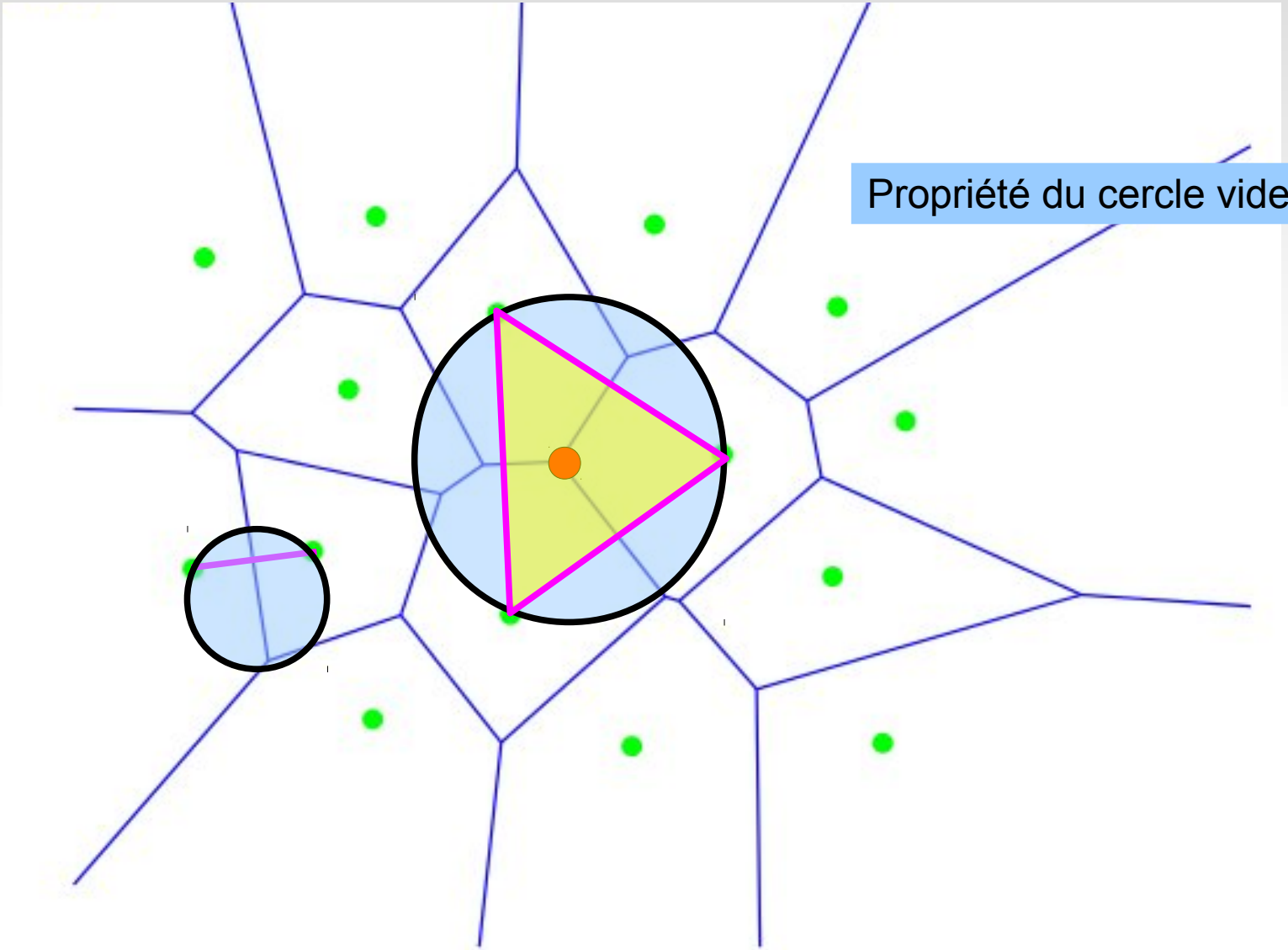
Remaillage

Planification de trajectoires

Voronoi - Delaunay



Propriété fondamentale



Algo1

Idée naïve : construire chaque cellule C_i de manière indépendante, comme intersection des n−1 demi-plans définis par les médiatrices des segments p_ip_j, $\forall j \neq i$

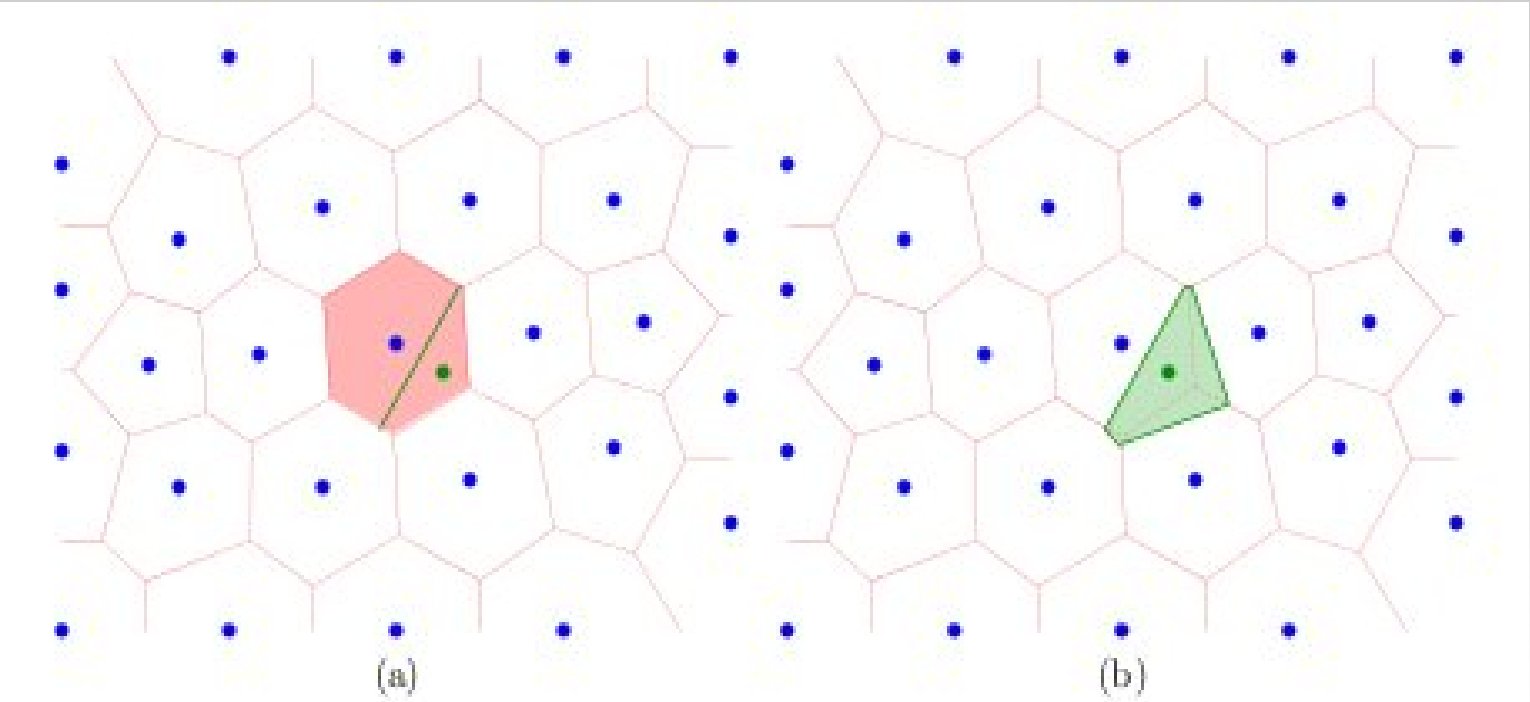
Construction duale de la construction de l'EC de (n−1) points du plan (le dual d'un point (x,y) est la droite d'équation b = xa − y et le dual d'une droite y = ax + b est le point (a,b))

EC calculée en O(nlog n).

Construction du diagramme de Voronoï en $O(n^2 \log n)$.

Voronoï: construction incrémentale

- Hyp : le diagramme de Voronoï d'un ensemble $P_{n-1}=\{p_1, \ldots, p_{n-1}\}$ de n−1 points du plan a été construit
- que modifier pour construire le diagramme de $P = P_{n-1} \cup \{p_n\}$?
- Intuitivement, modification locale.
- Les sommets de Voronoï supprimés à l'insertion = ceux dont le cercle associé (circonscrit aux germes voisins) contient p_n (l'intérieur des cercles centrés aux sommets de Voronoï et circonscrits aux germes voisins doit toujours être vide).
- Ces sommets sont dans une zone limitée du diagramme -> algorithme rapide en O(n) pour insérer p_n et mettre à jour le diagramme de Voronoï



Construction incrémentale du diagramme de Voronoï (a) Détection de la cellule C_i contenant p_n et construction de la médiatrice de p_i p_n .
(b) Construction itérative de la frontière de C_n .

- La cellule C_n peut avoir au pire n−1 arêtes ; le temps de calcul de cette cellule est donc en O(n).
- Puisqu'il y a n diagrammes de Voronoï successifs à calculer, la complexité totale de cet algorithme est en $O(n^2)$.

Algorithme de Green et Sibson

- 1. trouver le germe p_i tel que $p_n \in C_i$;
- 2. tracer la médiatrice du segment $p_i\ p_n$ et calculer ses intersections x_1 et x_2 (au plus 2 car C_i est convexe) avec la frontière de C_i
- 3. $x_1\ x_2$ = arête de Voronoï du diagramme de P (arête séparant C_i de C_n). x_2 est sur une arête de Voronoï du diagramme de P_{n-1} , séparant C_i d'une autre cellule C_j
- 4. recommencer le processus en remplaçant p_i par p_j

  5. itérer jusqu'à retomber sur x_1
- 6. on a ainsi la frontière de C_n ; mettre à jour les arêtes de Voronoï qu'elle intersecte, en supprimant les morceaux à l'intérieur de C_n .

Voronoï : diviser pour régner

- $O(n^2)$ pas optimal : algorithme en O(nlogn) de Shamos et Hoey en 1975
- jamais utilisé en pratique car trop compliqué à implémenter.
- stratégie “diviser pour régner” : P divisé en P_1 (points les plus “à gauche”) et P_2 (points les plus “à droite”) du plan ($\#P_1 = \#P_2$).
- Diagrammes de Voronoï de P_1 et P_2 calculés récursivement
- Fusion des deux diagrammes -> celui de P . Faite en temps linéaire, mais difficile.

Voronoï : algorithme par balayage

- Algorithme de Steven Fortune très utilisé en O(n log n)
- facilement implémentable.
- “balaie” progressivement le plan par une ligne et selon une certaine direction, de telle manière qu’à tout moment, dans la zone déjà balayée, le diagramme de Voronoï est construit de manière définitive.

Algorithme par Balayage Sweep Line Algorithm (Fortune)

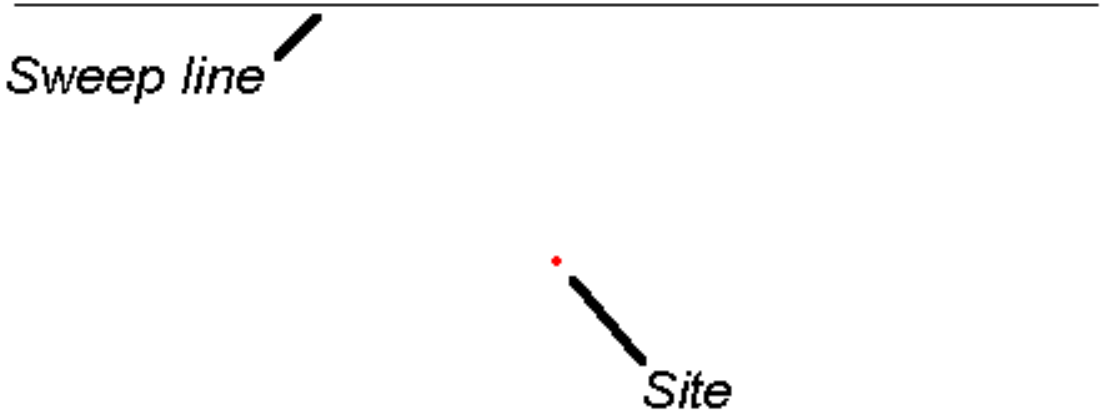
O(n log n)

Principe : partage le domaine du problème en 2 régions :

- Une région explorée
- Une région non explorée

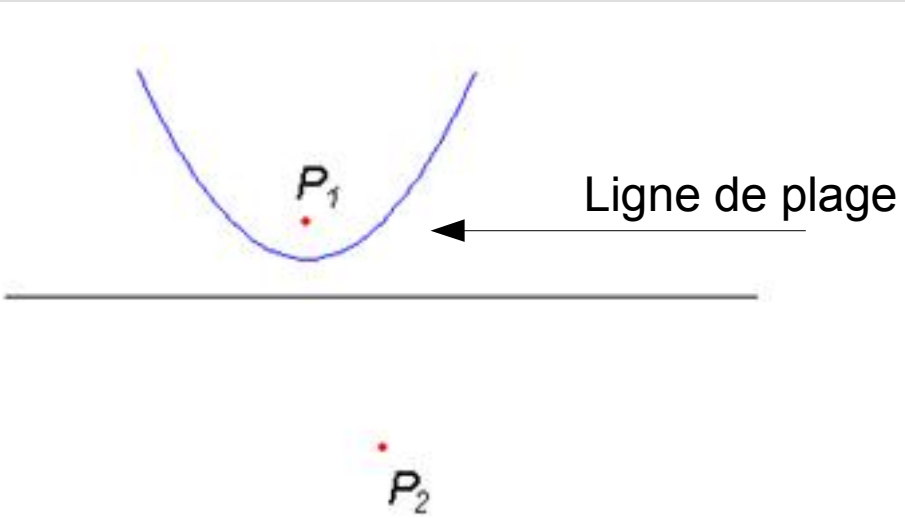
Principe

Une ligne de balayage S parcourt le domaine.
Avant de rencontrer le premier site, pas de question à se poser

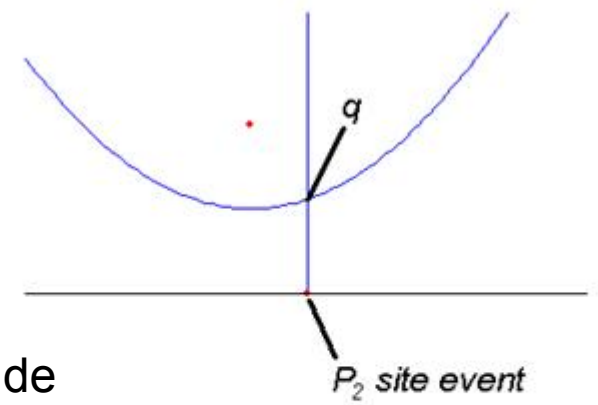


Quand S dépasse p1, quelle zone doit figurer dans la cellule de Voronoi de p1?
Comme il peut y avoir un nouveau site sous S on peut seulement dire que tout point plus proche de p1 que de S doit être dans la cellule de Voronoi de p1 à ce moment.

2e site

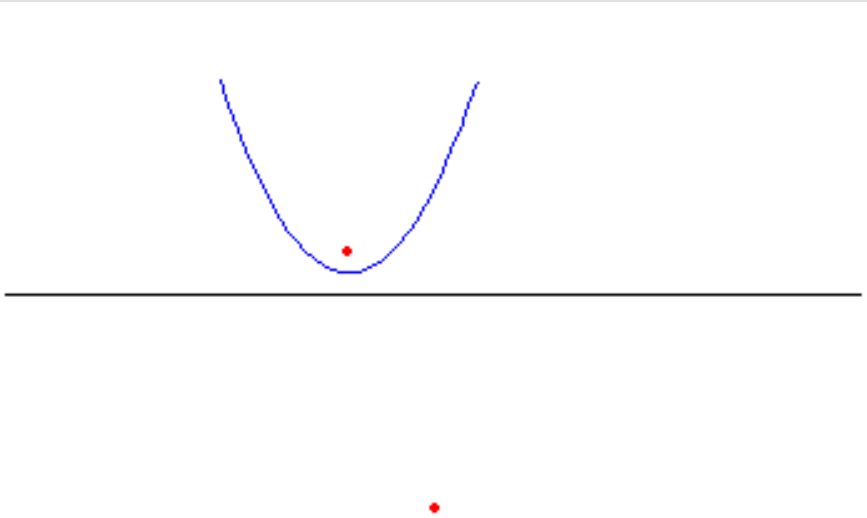


Evènement « site » : quand S rencontre un site.
Cela forme une parabole dégénérée qui coupe la parabole de p1 en un point q.
q est équidistant de p1 et p2, car il est sur la ligne de plage de p1 qui est le lieu des points à égale distance de p1 et de S.



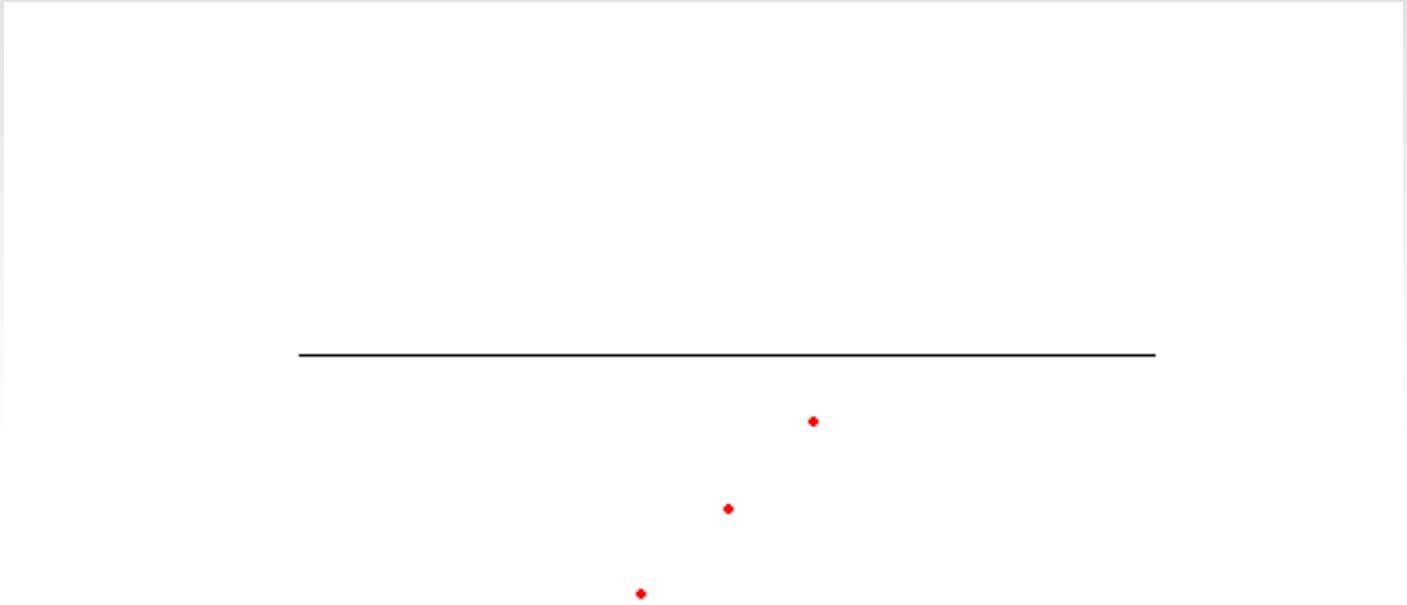
De plus, aucun site ne peut être plus proche de q sinon ligne de plage n'existerait pas en ce point. Donc ce point doit être sur l'arête de Voronoi qui sépare les 2 cellules définies par p1 et p2.

Ligne de plage



La ligne de plage est constituée de 3 arcs – la parabole de p2 et celle de p1 divisée en 2 par p2. La ligne de plage est toujours définie par la parabole la plus proche de S

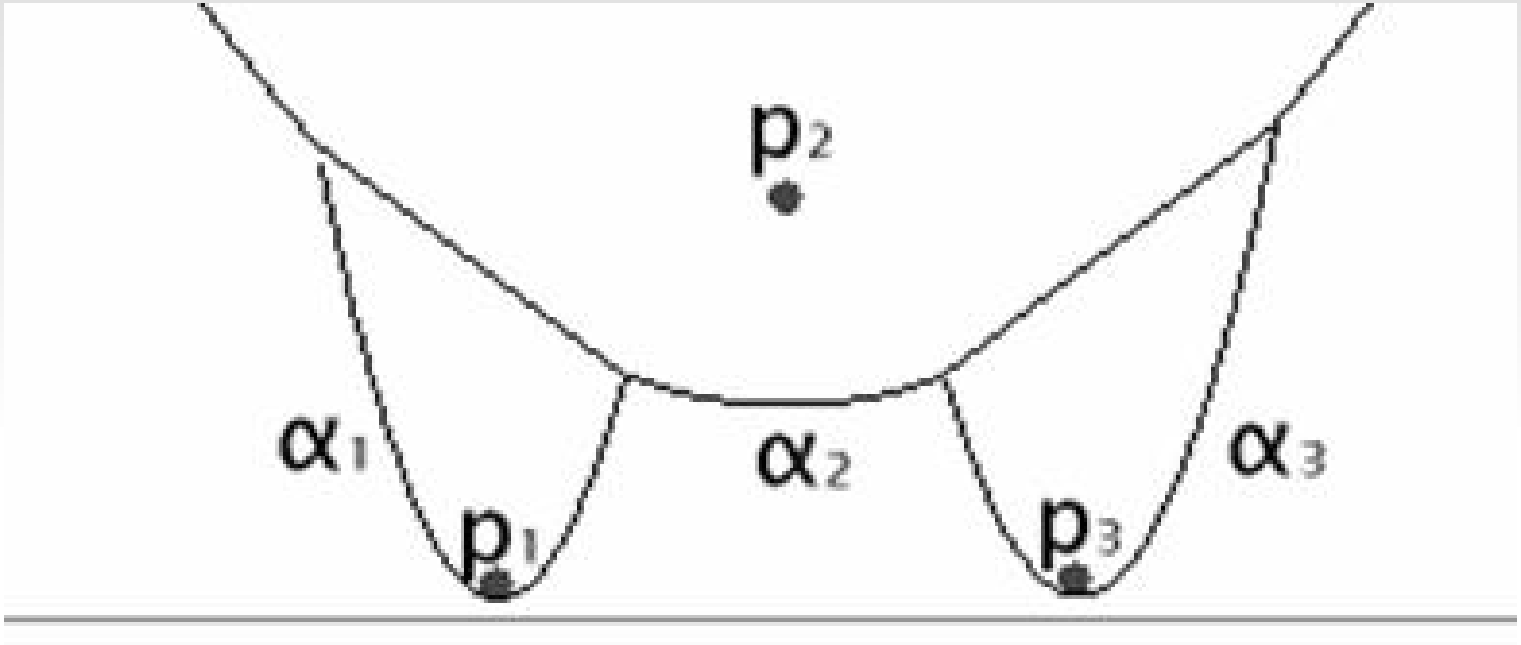
Arêtes de Voronoï



Au delà de 2 sites, chaque nouvel événement site crée un nouvel arc et définit une nouvelle arête de Voronoi

Important: un événement site est SEUL capable de produire un nouvel arc sur la ligne de plage. Essentiel pour l'algorithme.

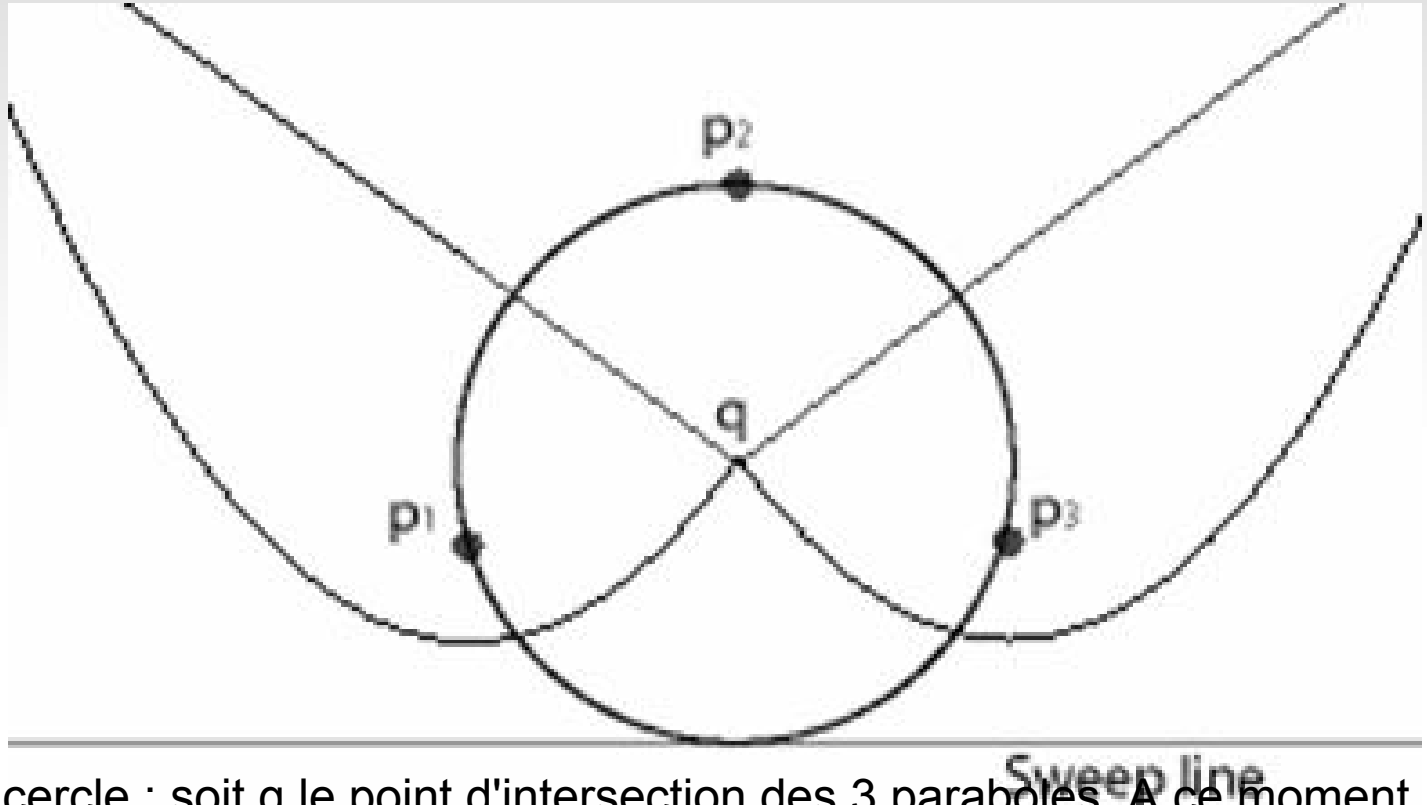
Disparition d'un arc



Au fur et à mesure que de nouveaux arcs naissent sur la ligne de plage, les anciens arcs éloignés de S disparaissent. Quand?

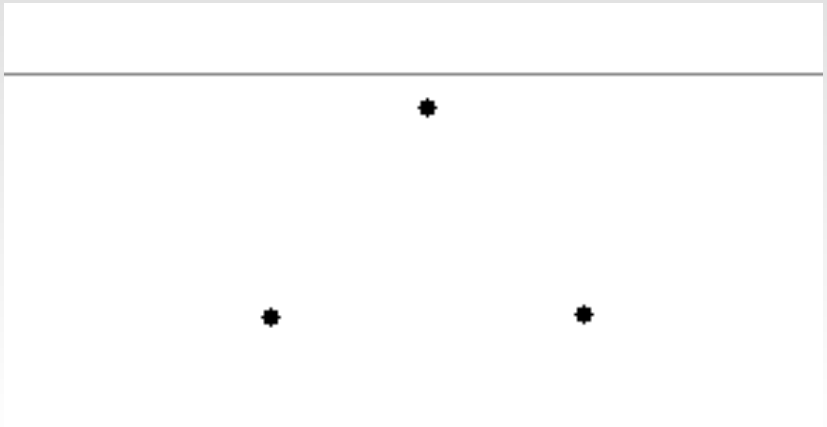
La ligne de plage est constituée de 3 arcs, α1, α2 et α3 générés par p1, p2 et p3 respectivement. Quand S descend, α2 est « entouré » par ses 2 voisins et $\alpha 2$ se réduit à un point.

Événement Cercle



Événement cercle : soit q le point d'intersection des 3 paraboles. A ce moment, on peut tracer un cercle passant par p1, p2 et p3 et le plus proche point sur S centré en q. Ce cercle existe parce que chaque site est équidistant de q et de S. De plus, il ne peut y avoir un autre site intérieur à ce cercle puisqu'alors q serait plus proche de ce nouveau site que de S, contredisant le fait que q est sur la ligne de plage.

Sommet de Voronoï



Ce point sera un sommet dans le diagramme de Voronoi en construction. Les 2 points d'arrêt qui définissent les arêtes de Voronoï convergent en q, laissant un seul point d'arrêt, qui définit un sommet de Voronoi de degré 3. Pour les sommets de Voronoi de degré supérieur à 3, le même processus intervient, sauf que 2 arcs au moins disparaissent en un seul point.

Important: un arc ne peut disparaître de la ligne de plage que lors d'un événement cercle

Algorithme de Balayage
Sweep Algorithm

- On ne calcule pas les paraboles définissant la ligne de plage (trop long et inutile).
- Calculs seulement lors des changements de topologie de la ligne de plage :
  - événement « site » quand un site rencontre S
  - Événement « cercle » quand un arc disparaît

Structures de données

1) Une file de priorité pour les événements site et cercle

· Tient à jour les changements de topologie. La priorité est déterminée par le balayage de S. Par exemple, si S part des y max et descend, l'événement site a une priorité plus grande s'il a un y plus grand. Idem pour les événements cercle. Dans l'exemple précédent, la priorité dépend du y de S quand S est tangent au bas du cercle correspondant à l'événement cercle.

· L'événement site enregistre la position du site. Les événements cercle enregistrent leur position, ainsi qu'un pointeur sur l'arc (une feuille) dans l'arbre binaire qui disparaît quand l'événement cercle survient.

· L' algorithme progresse en dépilant les événements et en calculant le changement de topologie de la ligne de plage.

· Les événements site sont connus en amont et peuvent être empilés à l'initialisation. Les événements cercle sont empilés au changement de topologie.

2) Une liste des sommets des arêtes pour mémoriser le diagramme de Voronoi en construction

· Quand un événement site définit un nouvel arc parabolique sur la ligne de plage, on commmence à construire une nouvelle arête de Voronoi.
Quand un événement cercle intervient et que 2 (ou plus) arêtes convergent en un point, cela devient un nouveau sommet de Voronoi, et une nouvelle arête de Voronoi est issue de ce nouveau point d'arrêt. La liste est utilisée pour tenir à jour ces arêtes et sommets.

· A la fin de l'algorithme la liste contient le diagramme de Voronoi.

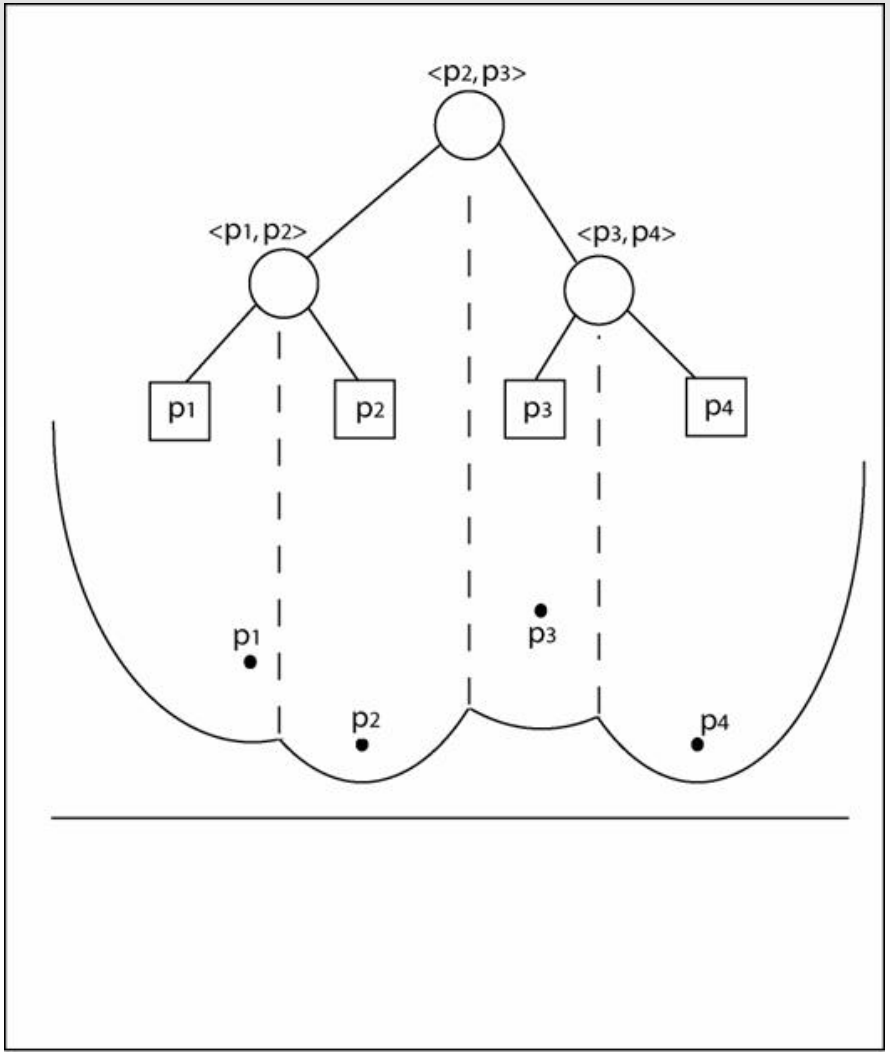
Un arbre de recherche binaire équilibré (ARBE) pour tenir à jour la topologie de la ligne de plage

· Chaque feuille de l'arbre stocke un site qui définit un arc. Chaque nœud stocke les 2 sites qui définissent les points d'arrêt sur la ligne de plage. Il est aisé de déterminer où un événement site affecte la ligne de plage. Dans notre exemple, on comparera la coordonnée x de chaque nouveau site pour trouver la position sur la ligne de plage. C'est en O(logn) à cause de l'ARBE.

· Un nouvel arc divise un arc existant en 2 : on supprime la feuille représentant l'arc original et on la remplace par un sous-arbre à 3 feuilles. La feuille centrale contient le nouveau site ajouté, et les 2 autres le site original. Les nœuds internes sont mis à jour pour indiquer les changements de points d'arrêt. L'arbre est rééquilibré si nécessaire.

· De plus chaque feuille stocke un pointeur vers un événement cercle dans la liste de priorité où l'arc défini par ce site disparaîtra. Si l'événement n'existe pas encore ou si le cercle n'existe pas le pointeur est à NULL. Un événement cercle est calculé en comparant les points d'arrêt de chaque côté de l'arc. S'il y a 0 ou 1 point d'arrêt, il n'y a pas d'événement cercle. S'il y en a 2, un test détermine s'ils convergent. Si oui, alors c'est un événement cercle potentiel qui est ajouté à la liste de priorité.

ARBE



Algorithme

Main(liste des sites)

Structures de données initialisées. Insérer les événements site dans la liste de priorité selon leur coordonnée y.

```
While(Queue non vide)
    event = dépiler premier evenement de la liste
    If(event = = Site Event)
                ProcessSite(event)
    Else
                ProcessCircle(event)
Finir toutes les arêtes dans l'ARBE
```

ProcessSite(event)
    If (ARBE = = null)
        Racine = event
    Else
        Trouver la position correcte dans ARBE cet événement (en partant de la racine et en testant les points d'arrêt des nœuds internes. Prendre fils droit ou gauche selon que la corrdonnée x de l'événement courant est supérieure ou inférieure à celle du point d'arrêt ; si égale, indifférent. Stop quand une feuille est rencontrée.)
        Si l'arc intercepté par le site contient un pointeur vers un événement cercle, supprimer cet événement. Supprimer aussi les événements cercle liés à cet arc. Ces événements sont les voisins de l'arc dans l'ARBE (ils n'existent pas toujours)
        Supprimer la feuille de cet événement site et la remplacer par un sous-arbre à 3 feuilles (g et d pour site original, centrale pour le nouveau. Les 2 nœuds internes représentent les 2 points d'arrêt créés avec la nouvelle arête. Créer une nouvelle arête dans la liste des sommets-arêtes et faire pointer chaque nœud interne vers les extrémités.
        Tester si des événements cercle sont dûs à cet nouveau site (NB : on n'a pas besoin de tester les triplets de feuilles où la centrale est le nouveau site car les points d'arrêt ne peuvent converger. Par contre il faut tester les triplets où le nouveau site est à gauche ou à droite. Si les points d'arrêt convergent, calculer la prioorité de l'événement cercle et le metrre dans la liste. Créer un pointeur de la feuille centrale vers cet événement.

ProcessCircle(event)

Mettre à jour les points d'arrêt liés à l'arc qui disparaît. Les arêtes sur lesquels les ponts d'arrêt pontent sont finies.

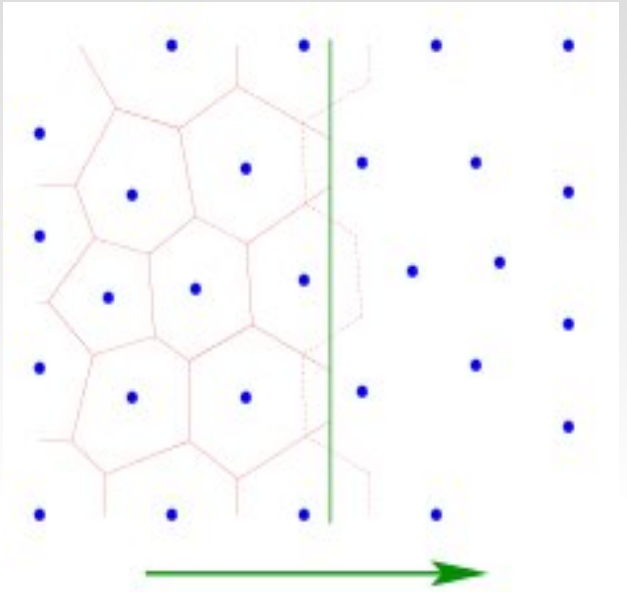
Tester les événements cercle liés à cet arc dans les arcs immédiatement à gauche et à droite sur la ligne de plage. S'il y a des événements cercle en ces nœuds, les détruire.

Supprimer la feuille correspondant à l'arc dans l'ARBE. Supprimer le père de ces nœuds internes. Mettre à jour les points d'arrêt crées dans l'ARBE. 2 points d'arrêt disparaissent lors de cet événement, et un nouveau est crée. Ce dernier doit pointer sur une extrémité de la nouvelle arête dans la liste des arêtes. L'autre extrémité contient le sommet crée par cet événement.

Tester les nouveaux triplets d'arcs créés par réarrangement de l'ARBE pour les événements cercle. Si un cercle est détecté, le mettre dans la liste, et mettre les pointeurs des feuilles qui disparaissent dans cet événement.

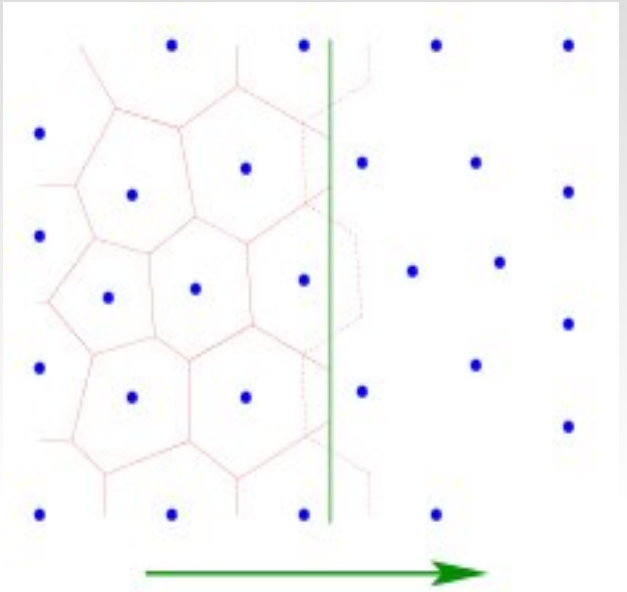
Voronoï : algorithme par balayage

- Impossible ? l'apparition d'un nouveau point de P lors du balayage va modifier certaines cellules de Voronoï avant ce point
- L'idée : passer en 3D afin d'"anticiper" les modifications du diagramme

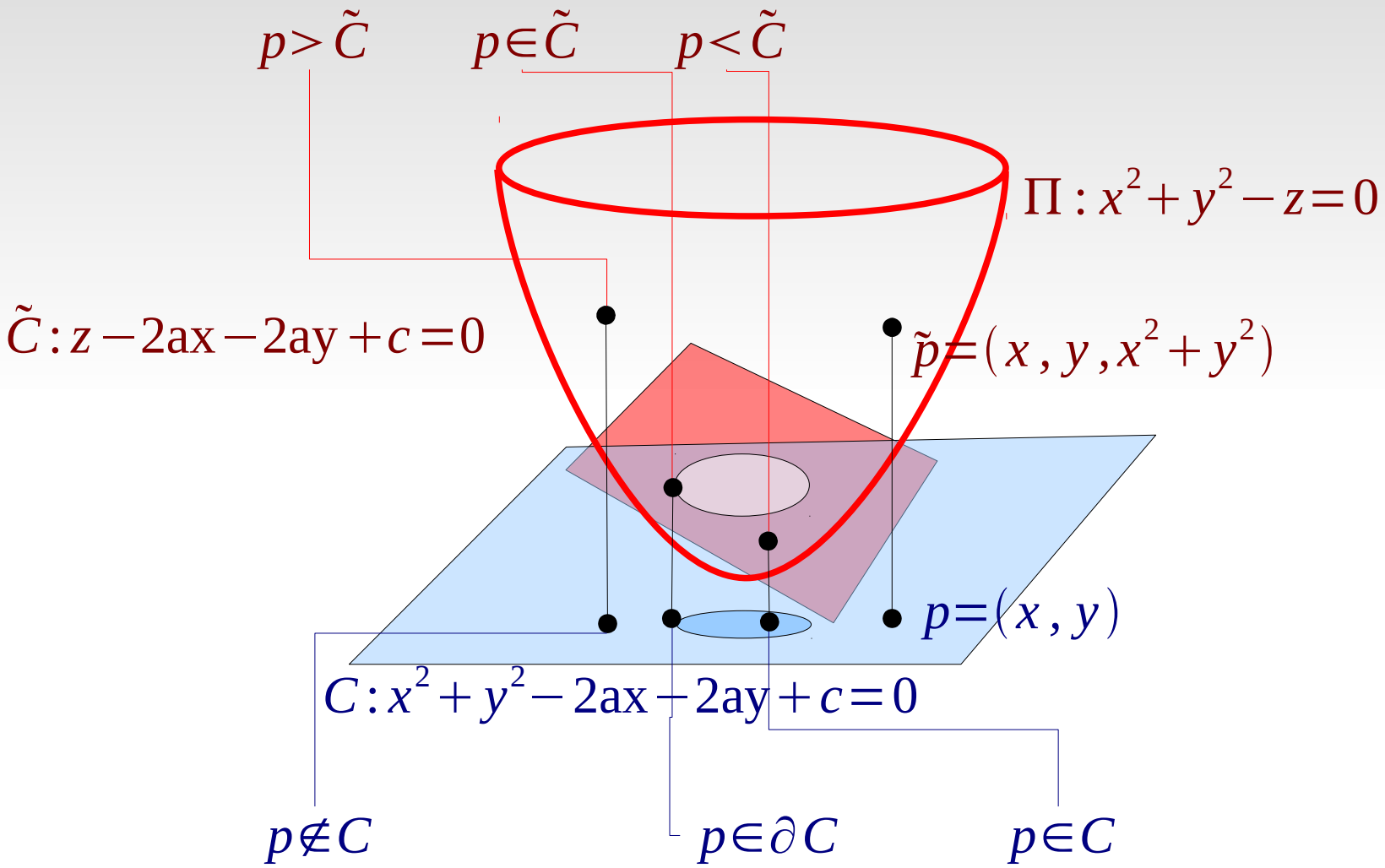


Voronoï : algorithme par balayage

- Impossible ? l'apparition d'un nouveau point de P lors du balayage va modifier certaines cellules de Voronoï avant ce point
- L'idée : passer en 3D afin d'"anticiper" les modifications du diagramme

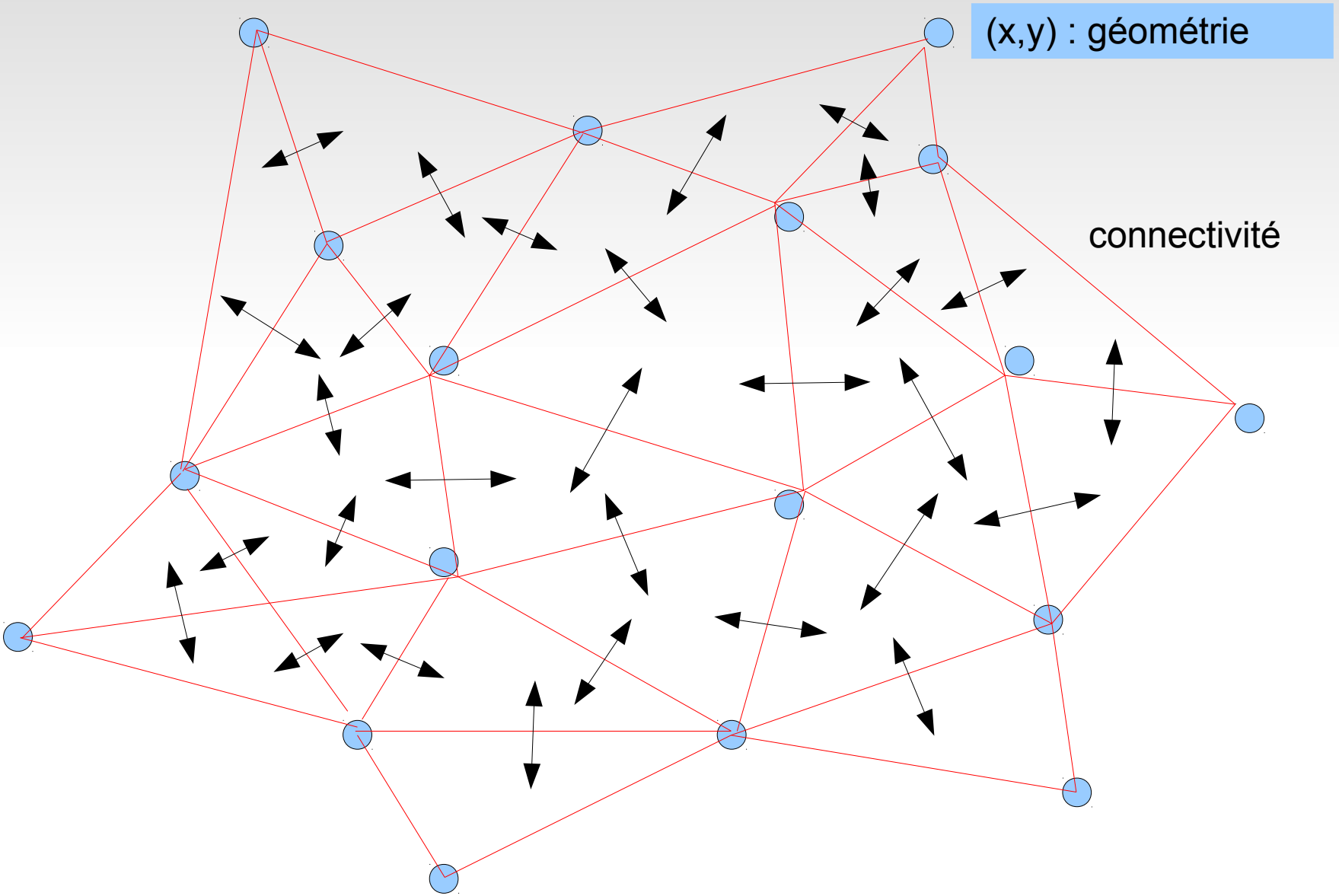


Algorithme d'élévation



Cocyclicité <=> Coplanarité

Quelques structures de données

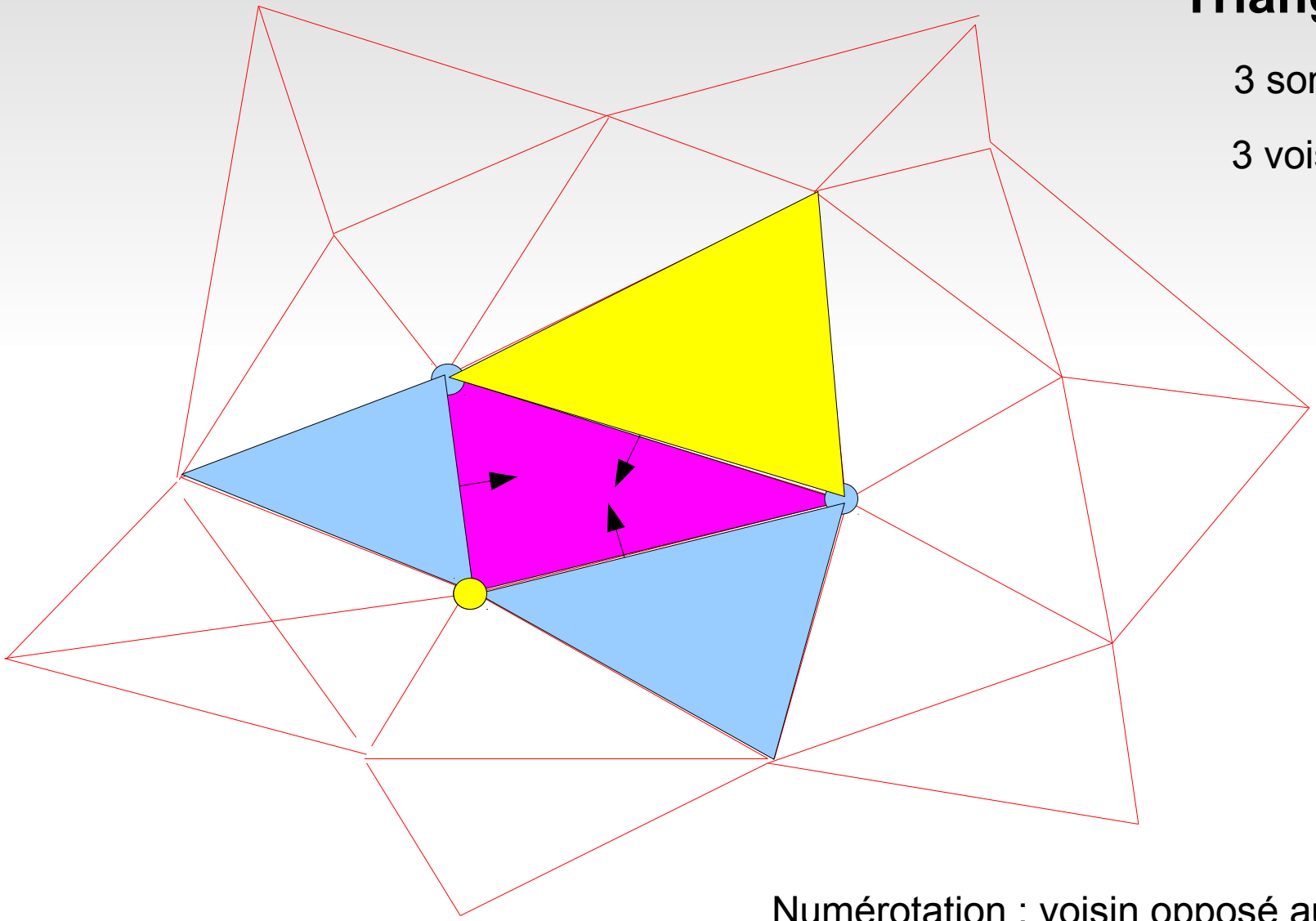


Représentation basée triangles

Triangle =

3 sommets

3 voisins



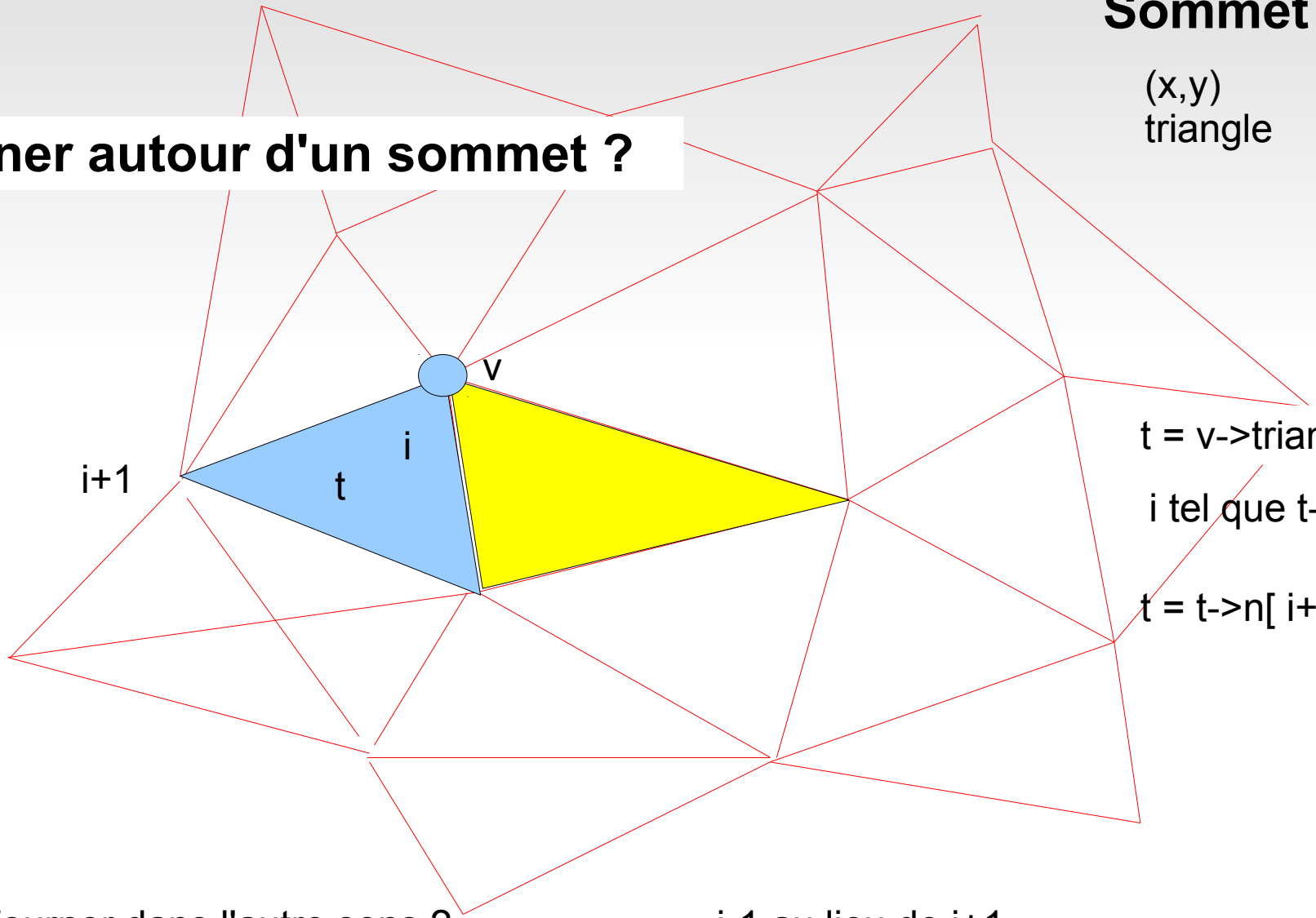
Numérotation : voisin opposé au sommet

Représentation basée triangles

Sommet =

(x,y)
triangle

Tourner autour d'un sommet ?



t = v->triangle;

i tel que t->v[i] == v;

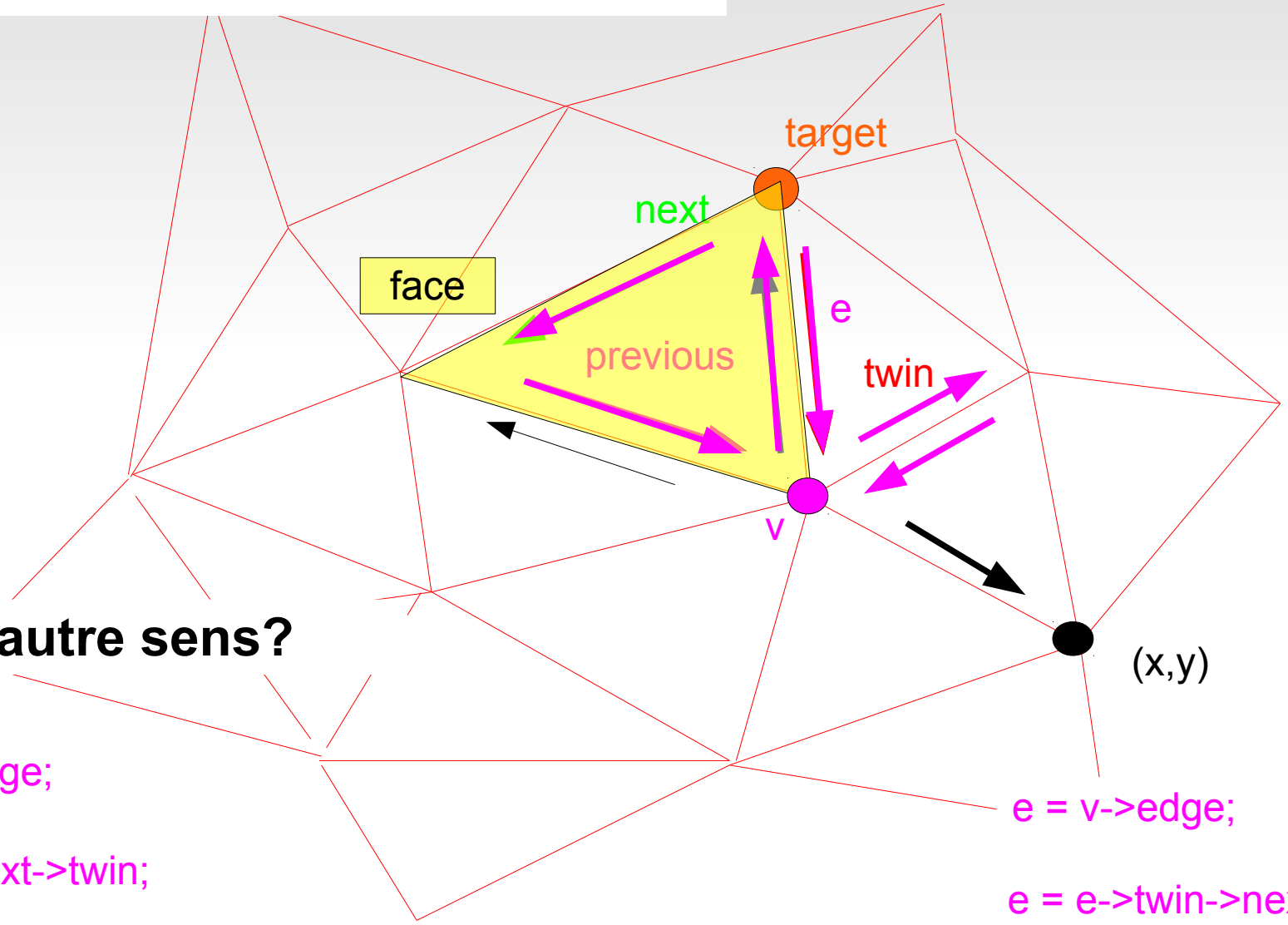
t = t->n[i+1 mod 3];

Tourner dans l'autre sens ?

i-1 au lieu de i+1

Demi-arêtes

Tourner autour d'un sommet?



Dans l'autre sens?

e = v->edge;

e = e->next->twin;

e = v->edge;

e = e->twin->next->next;