De la géométrie algorithmique au calcul géométrique : l'exemple de la triangulation de Delaunay

Olivier Devillers

16 juillet 2004

Plan indicatif du cours

- **Cours 1 : Introduction à la géométrie algorithmique.** Panorama des problèmes et techniques (Enveloppes convexes, Triangulations, Arrangements, Enveloppes inférieures, Reconstruction, Maillages, Randomisation, Robustesse, Objets courbes.)
  Quelques résultats sur enveloppes convexes.
- **Cours 1bis : Introduction à la bibliothèque CGAL** Un peu de C++.
  Standard Template Library.
  Computational Geometry Algorithms Library.
- **Cours 2 : Triangulation de Delaunay, premières propriétés et algorithmes.** Voir chapitre 3.
- **TD 2 :** Prise en main CGAL. TD assurés par Olivier Devillers et Steve Oudot.
  http ://www.cgal.org
- **Cours 3 : Calculer Delaunay, grand classique de la géométrie algorithmique.** Voir chapitre 4 en particulier paragraphes 4.1 et 4.3.
- **TD 3 :** Énumérer les plus proches voisins.
- **Cours 4 : La randomisation.** Voir paragraphes 4.5 et 8.3.
- **TD 4 :** Benchs sur Delaunay avec différents ordres d'insertion.
- **Cours 5 : De la théorie à la pratique : la robustesse de algorithmes.** Voir chapitres 5, 6 et 8 en particulier paragraphes 5.1.
- **TD 5 :** Robustesse (inconsistance, debug) bench, arithmétiques.
- **Cours 6 : Généralisations**
  Diagramme de puissance.
  Delaunay contraint...
- **TD 6 :** Triangulation contrainte.
- **Cours 7 : Application à la reconstruction tridimensionnelle.** Voir chapitre 13.
- **TD 7 :** Alpha formes.
- **Cours 8 : Application aux maillages**
- **TD 8 :** Maillage
- **Cours 9 : Autres problèmes en géométrie algorithmique**
  Enveloppes inférieures (Davenport Schinzel)
  Arrangements (théorème de la zone 2D, Algorithme de Bentley Ottmann).
- **TD 9 :** Lloyd en 2D
- **Projets** Une liste définitive de sujets vous sera fournie. Voici quelques

indications sur les sujets possibles.
- Crust
- Surfaces de subdivision
- Estimation de courbure
- Paramétrisation
- Simplification
- Triangle strips
- Lissage (laplacien)

Avertissement aux étudiants et autres lecteurs

- Ce document est encore très imparfait et je m'en excuse.
- Pour combler ses lacunes,il y a de nombreux ouvrages sur la géométrie algorithmique qui existent déjà, vous pourrez les consulter pour avoir plus de précisions sur certains sujets (par exemple [9, 46, 15, 25, 40, 45]).
- L'originalité de ce document est dans l'angle de vue «pratique» et l'insistance sur trai certains aspects liés à la mise en œuvre des algorithmes géométriques, sujet plus récent et moins traité dans les livres.
-
- Je suis ouvert à toutes critiques ou discussions sur la qualité de ce document afin de l'améliorer pour l'avenir. N'hésitez pas à me faire part de vos remarques, tant sur les détails de rédaction que sur la présentation d'ensemble ou tout aspect que vous jugerez utile.
- Signalez moi tout ce qui est affirmé dans ce document et qui selon vous mériterai des explications supplémentaires.
- Quand il est écrit «`(paragraphe à finaliser :=)` », il est probable que vous n'aurez rien de mieux cette année. Reportez vous aux références indiquées.

Table des matières

Chapitre 1

Présentation

Le but de ce document est de présenter les grandes tendances de la *géométrie algorithmique* actuelle, et en particulier son évolution vers ce que nous appellerons le *calcul géométrique*.

Née dans les années 1970, la géométrie algorithmique cherche à concevoir et analyser des algorithmes permettant de calculer ou manipuler des objets géométriques, c'est un domaine scientifique à part entière qui entretient des relations privilégiées avec la topologie, les géométries algébrique et combinatoire, et l'algorithmique. Les domaines d'applications sont nombreux, citons la CAO, la robotique, la vision par ordinateur, l'informatique graphique, l'analyse numérique, la cartographie, la chimie, la botanique.

Plutôt que d'inventorier les grands problèmes du domaine que sont enveloppes convexes, arrangements, structures pour requêtes, triangulations... nous allons choisir un problème géométrique précis : la triangulation de Delaunay, et expliquer sur ce problème particulier les différentes solutions proposées par la géométrie algorithmique. Nous avons choisi la triangulation de Delaunay parce que c'est une des structures les plus populaires de par ses nombreuses applications (modèles de terrains, maillages, reconstruction de formes...) et que la plupart des grands principes que nous voulions illustrer trouvent une instance dans la triangulation de Delaunay.

Les premiers explorateurs de la géométrie algorithmique cherchaient essentiellement à concevoir des algorithmes ayant une bonne complexité. Ils entendaient par là une bonne complexité théorique lorsque le volume de donnée à traiter était très grand, et ceci quelque soit la disposition de ces données. Ces algorithmes reposaient également sur deux hypothèses complémentaires : la première est que l'on dispose d'une arithmétique exacte sur les nombres réels (au sens mathématique) la seconde qu'il n'y a pas de cas dit dégénérés, c'est à dire que trois points ne sont jamais alignés quatre points ne sont jamais cocycliques...

L'hypothèse de non dégénérescence permet essentiellement de simplifier la présentation des algorithmes en s'affranchissant d'une description des cas particuliers. Mais elle a des effets de bord malheureux : tout d'abord, elle fait négliger

le traitement de ces cas particuliers qui après avoir été omis dans l'algorithme théorique risquent d'être omis au moment de la programmation, ensuite, ces hypothèses sont parfois utilisées dans l'analyse du temps de calcul et de l'occupation mémoire, et un algorithme de bonne complexité en situation générale peut avoir un comportement catastrophique dans une situation dégénérée.

Quant à l'hypothèse de calcul sur les nombres réels, elle est extrêmement forte et a des conséquences désastreuses lorsque l'on cherche à programmer effectivement l'algorithme. L'utilisation de l'arithmétique flottante de la machine comme approximation du calcul sur les nombres réels a généralement pour conséquence de prendre des décisions incohérentes dans les cas litigieux (points presque alignés par exemple) et de conduire l'algorithme dans des situations non prévues car impossible géométriquement. Dans le meilleur des cas le programme plante, mais il peut aussi calculer n'importe quoi ou boucler, selon les implantations.

Les développements récents en calcul géométrique on pour but d'apporter des réponses à la question *comment passer de la géométrie algorithmique à des algorithmes effectivement programmés ?* Parmi les grands points abordés on citera : comment gérer les problèmes de précision numérique (arithmétique sur les réels), comment gérer les cas dégénérés, et quelle est la vrai complexité d'un algorithme et non pas la complexité asymptotique dans le cas le pire. Tous ces points seront abordés à travers l'exemple de la triangulation de Delaunay.

Chapitre 2

Généralités

Avant d'aborder la triangulation de Delaunay, nous n'échapperons pas à quelques considérations générales et définitions concernant la géométrie algorithmique.

2.1 La géométrie algorithmique calcule de la combinatoire

La géométrie algorithmique classique s'intéresse aux structures combinatoires ayant une définition géométrique. C'est à dire, étant donné un ensemble fini $\mathcal{S}$ de données géométriques (points, segments, cercles...) on regarde le résultat comme un objet combinatoire construit sur $\mathcal{S}$. Par exemple, le résultat cherché peut être une partie de $\mathcal{S}$, une partie de $\mathcal{S}^k$ ou un graphe ayant ses sommets dans $\mathcal{S}$... Il faut bien insister sur le fait que le résultat est plus combinatoire que géométrique. Par exemple on peut chercher les points d'intersections dans un ensemble de segments du plan sous la forme de l'ensemble des paires de segments qui se coupent et non pas sous la forme des coordonnées effectives de ces points d'intersections.

Un autre exemple est l'enveloppe convexe de points. Étant donné $\mathcal{S}$ un ensemble de points du plan, on appelle enveloppe convexe de $\mathcal{S}$, $\mathcal{CH}(\mathcal{S})$, l'ensemble des barycentres à coefficients positifs de points de $\mathcal{S}$.

$$\mathcal{CH}(\mathcal{S}) = \{x \ ; \ \forall p \in \mathcal{S} \exists \alpha_p \geq 0, x = \sum p \in \mathcal{S} \alpha_p p\}.$$

Il est facile de voir que $\mathcal{CH}(\mathcal{S})$ est un polygone dont les sommets sont des points de $\mathcal{S}$, $\mathcal{CH}(\mathcal{S})$ peut donc être donné par la liste (ordonnée) des points de $\mathcal{S}$ qui forment ses sommets. On retrouve donc un objet combinatoire, la liste des sommets, comme représentation d'un ensemble infini de points $\mathcal{CH}(\mathcal{S})$.

Étant donné la définition d'une telle structure, l'objet de la géométrie algorithmique est d'une part l'analyse de la complexité de cette structure : quelle est la taille du résultat en fonction du nombre de données, et d'autre part,

la conception et l'analyse des algorithmes géométriques permettant de calculer effectivement ce résultat.

Nous considérons donc un algorithme géométrique comme produisant un résultat combinatoire, tel que la liste des sommets de l'enveloppe convexe, plutôt qu'un résultat géométrique, comme le sous ensemble du plan formé par l'enveloppe convexe. Le point de vue combinatoire apporte une nature essentiellement discontinue au problème. Si l'on déplace les points de $\mathcal{S}$ la liste des sommets de l'enveloppe convexe change seulement à certains «évènements», même si l'enveloppe convexe varie de manière continue dans le plan. Un tel algorithme a donc une nature essentiellement discontinue, même si le sous ensemble du plan *enveloppe convexe* varie de manière continue.

2.2 Prédicats géométriques

Un algorithme géométrique est donc un processus essentiellement combinatoire durant lequel un certain nombre de décisions sont prises. Ces décisions reposent sur des questions à caractère géométrique, en général assez simple ; par exemple l'orientation d'un triangle. Ces tests géométriques sont appelés prédicats.

Un *prédicat géométrique* est une question géométrique qui à partir d'un nombre fini (borné) d'objets admet un nombre fini (borné) de réponses.

Par exemple :

- Orientation d'un triangle.
  trois points $\longmapsto$ direct, indirect, colinéaire.
- Orientation d'un simplex en dimension d.
  $d+1$ points $\longmapsto$ direct, indirect, coplanaire.
- Cocircularité.
  4 points $\longmapsto$ le dernier point est intérieur, extérieur, sur le cercle passant par les trois premiers points.
- Intersection.
  Deux courbes $\longmapsto$ les courbes se coupent transversalement, sont tangentes, sont disjointes.
- Intersection.

Un cercle C et un segment[a] de droite $[pq]$ $\longmapsto$
- $|(pq) \cap C| = 2$
- $|(pq) \cap C| = 1$ et pq tangent à C
- $|(pq) \cap C| = 1$, p à l'intérieur et q à l'extérieur de C
- $|(pq) \cap C| = 1$, q à l'intérieur et p à l'extérieur de C
- $|(pq) \cap C| = 1$, $p \in C$ et q à l'extérieur de C
- $|(pq) \cap C| = 1$, $q \in C$ et p à l'extérieur de C
- $|(pq) \cap C| = 0$ et $p, q \in C$
- $|(pq) \cap C| = 0$, $p \in C$ et q à l'intérieur de C
- $|(pq) \cap C| = 0$, $p \in C$ et q à l'extérieur de C
- $|(pq) \cap C| = 0$, $q \in C$ et p à l'intérieur de C
- $|(pq) \cap C| = 0$, $q \in C$ et p à l'extérieur de C
- $|(pq) \cap C| = 0$ et p, q à l'intérieur de C
- $|(pq) \cap C| = 0$ et p, q à l'extérieur de C

[a] $[pq]$ désigne le segment fermé et (pq) le segment ouvert ne contenant pas ses extrémités.

L'évaluation de prédicats géométriques revient en général à l'évaluation du signe d'une certaine expression, par exemple pour l'orientation du triangle pqr on cherche le signe de $\begin{vmatrix} 1 & 1 & 1 \\ x_p & x_q & x_r \\ y_p & y_q & y_r \end{vmatrix}$. Ce signe peut être positif pour un triangle direct, négatif pour un triangle indirect et nul si les points sont alignés. Lorsque l'expression est nulle, on dit que les données sont en position dégénérée.

Un prédicat plus compliqué reste en général une combinaison d'évaluation de signes d'expressions.

2.3 Complexité

On appelle complexité d'un algorithme le nombre d'opérations élémentaires nécessaire à son exécution. En géométrie algorithmique, on considère généralement un prédicat comme une opération élémentaire et on essaye de compter le nombre d'évaluation des prédicats nécessaire soit dans le cas le pire des cas pour les données, soit avec certaines hypothèses de type probabilistes.

On parle également de borne inférieure d'une complexité, il s'agit du nombre minimal de prédicats à évaluer pour déterminer le résultat. On démontre en général de telles bornes inférieures en comptant le nombre de branchement nécessaire dans l'algorithme (*Information Theory Bound*), par exemple, trier n nombres nécessite au moins $n \log n$ comparaisons.

On se contente fréquemment des complexités asymptotiques, c'est à dire que l'on cherche juste à obtenir l'ordre de grandeur de la complexité lorsque le nombre de données est très grand. On parle alors, par exemple, de complexité en $O(n \log n)$ ce qui sigifie que pour n assez grand, il existe une certaine constante α tel que le nombre d'opérations décessaire soit plus petit que $\alpha n \log n$.

2.4 Configuration générale

Comme on l'a dit ci dessus, un algorithme repose sur l'évaluation de prédicats. On dira qu'un ensemble de données est en configuration générale pour un certain ensemble de prédicats, si il n'existe pas de sous-ensemble de ces données tels qu'un des prédicats soit dégénéré.

Par exemple, un ensemble de points du plan sera en position générale pour les prédicats d'orientation et de cocircularité si il n'y a ni trois points alignés ni quatre points cocycliques.

Chapitre 3

Triangulation de Delaunay

3.1 Définitions et propriétés

3.1.1 Diagramme de Voronoï

Le diagramme de Voronoï est une structure géométrique permettant de déterminer le plus proche voisin d'une requête. Étant donné un ensemble $\mathcal{S}$ de *sites* et une requête q on cherche à répondre à la question : «quel est l'élément de $\mathcal{S}$ le plus proche de q ?»

Dans le cas le plus simple, $\mathcal{S} = \{p_j\}$ est un ensemble de points du plan, q est également un point du plan, et plus proche fait référence à la distance euclidienne usuelle.

On appelle polygone de Voronoï V_i du point $p_i \in \mathcal{S}$ l'ensemble des points du plan plus proche de p_i que d'un autre point de $\mathcal{S}$

$$V_i = \{q; \forall j \neq i | qp_i| \leq |qp_j|\};$$

en d'autre termes, V_i est la région du plan où la réponse à une requête de plus proche voisin sera p_i. La partition du plan formée par l'ensemble des polygones de Voronoï est appelée diagramme de Voronoï. La figure 3.1 montre un exemple de diagramme de Voronoï

Quelques premières remarques simples permettent de se familiariser rapidement avec la structure du diagramme de Voronoï. Le diagramme de Voronoï est un découpage du plan en différentes cellules, ces cellules sont des polygones formés d'arêtes et de sommets. Comme on l'a déjà remarqué, dans une cellule V_i le site le plus proche est p_i. À la frontière entre V_i et V_j, p_i et p_j sont a égales distances ; l'arête séparant V_i et V_j est portée par la médiatrice de p_i et p_j. Les sites correspondants aux régions incidentes à un sommet du diagramme de Voronoï sont tous à égale distance de ce sommet ; il y a en général trois cellules incidentes V_i, V_j et V_k et le sommet de Voronoï est donc le centre du cercle circonscrit au triangle $p_ip_jp_k$.

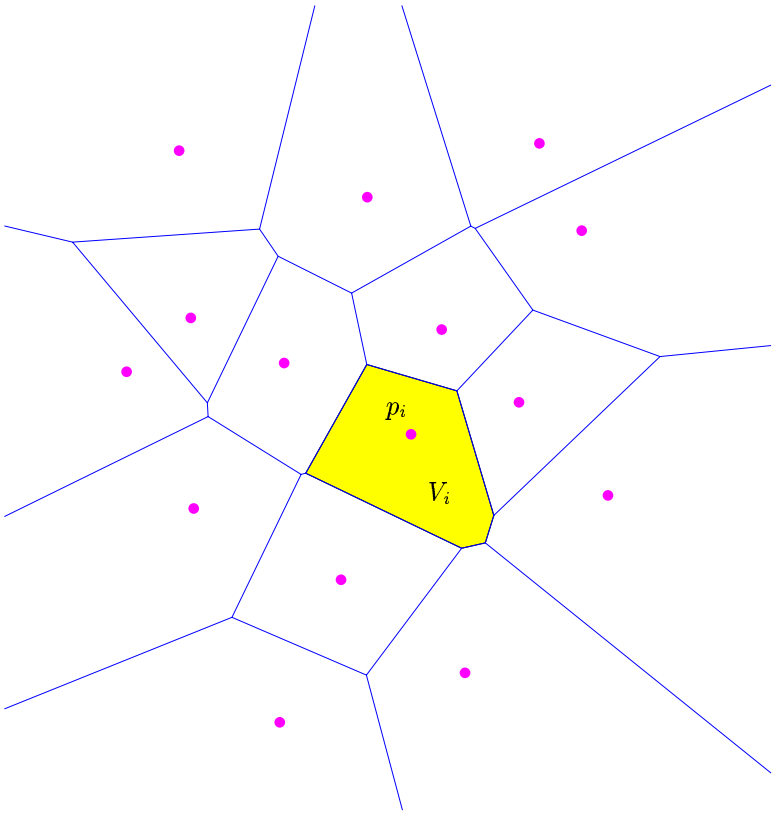

FIG. 3.1 – Exemple de diagramme de Voronoï

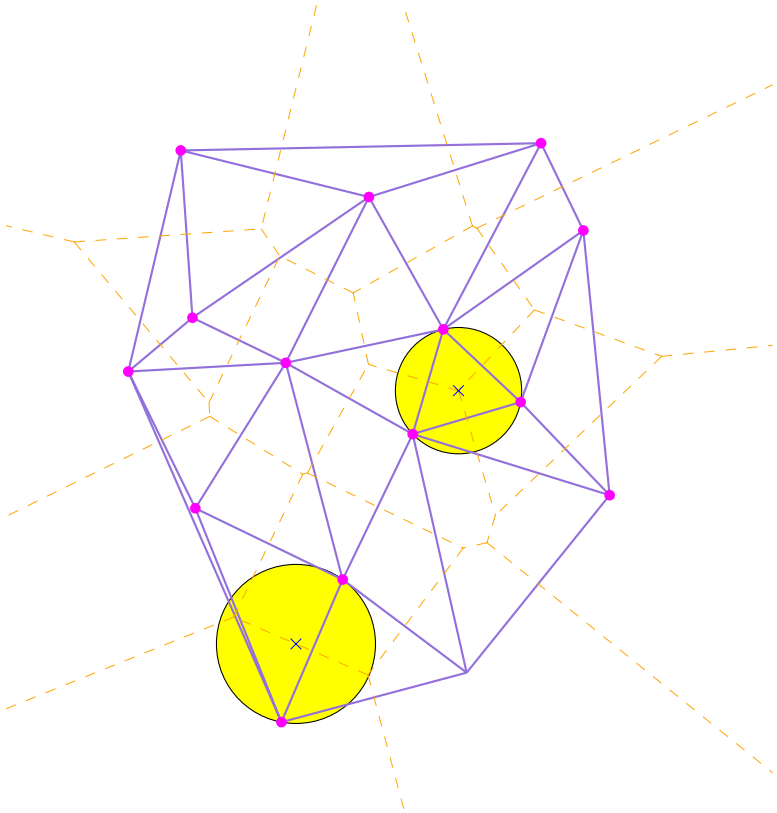

FIG. 3.2 – Exemple de triangulation de Delaunay.

3.1.2 Triangulation de Delaunay

La triangulation de Delaunay est classiquement définie comme le dual du diagramme de Voronoï. Si deux régions de Voronoï V_i et V_j sont voisines dans le diagramme de Voronoï, alors on relie p_i et p_j par une arête.

Si les points de $\mathcal{S}$ sont en position dite «générale», c'est-à-dire que les sommets du diagramme de Voronoï ne sont jamais incident à plus de trois cellules, l'ensemble des arêtes ainsi formées est appelé triangulation[1] de Delaunay de $\mathcal{S}$ et notée $\mathcal{DT}(\mathcal{S})$, un exemple est donné à la figure 3.2.

On déduit facilement de tout ce qui précède la propriété fondamentale de la triangulation de Delaunay :

Théorème 3.1 *$p_ip_jp_k \in \mathcal{S}$, $p_ip_jp_k$ est un triangle de la la triangulation de Delaunay de $\mathcal{S}$ si et seulement si le cercle circonscrit à $p_ip_jp_k$ ne contient aucun point de $\mathcal{S}$.*

[1] On appelle triangulation de $\mathcal{S}$ un ensemble d'arêtes reliant des points de $\mathcal{S}$ sans intersections (deux arêtes ne se coupent pas) et maximal (on ne peut pas rajouter d'arêtes).

Démonstration En effet le centre Ω de ce cercle est équidistant de pi, p_j et p_k et donc est un sommet de Voronoï si et seulement si il n'y a pas d'autre point de $\mathcal{S}$ plus près de Ω que pi, p_j et p_k. ■

De même on a

Lemme 3.2 *$p_ip_j \in \mathcal{S}$, p_ip_j est une arête de la la triangulation de Delaunay de $\mathcal{S}$ si et seulement si il existe un cercle passant par p_i et p_j ne contenant aucun point de $\mathcal{S}$.*

3.1.3 Premières propriétés

Complexité

La relation d'Euler s'applique à tout graphe planaire. Elle stipule que $c - a + s = 1$ où c est le nombre de cellules (non compris la cellule infinie), a est le nombre d'arêtes et s le nombre de sommets.

Dans une triangulation, toutes les cellules sont des triangles sauf la cellule infinie qui a k cotés, on peut en déduire que $2a = 3c + k$. La relation d'Euler devient $2c - 3c - k + 2s = 2$ soit $c = 2s - k - 2 < 2s$.

Le nombre de triangles de Delaunay (ou de sommets de Voronoï) est donc borné par $2s$ et celui d'arêtes de Delaunay (ou de Voronoï) par $3s$.

Maximise le plus petit angle

Dans une triangulation de $\mathcal{S}$, le nombre de triangles, et donc d'angles ne dépends pas du choix d'une triangulation particulière d'après la démonstration ci dessus qui est valable pour toute triangulation.

On peut donc utiliser les valeurs de ces angles pour trier les triangulations. À une triangulation de $\mathcal{S}$ nous allons associer le $3c$-uplet formé par les angles de la triangulation triés en ordre croissant. L'ordre lexicographique sur les $3c$-uplets permet donc d'ordonner les triangulations de $\mathcal{S}$.

Théorème 3.3 *La triangulation de Delaunay est maximale pour l'ordre lexicographique des angles.*

Démonstration La preuve du théorème 3.3 sera faite ci dessous comme un corrolaire du théorème 3.4. ■

En particulier si on ne regarde que le premier élément du $3c$-uplet, c'est à dire le plus petit angle de la triangulation, celui ci est maximisé par Delaunay.

Ce critère d'apparence un peu abstraite est en fait une formalisation d'une des raisons de la popularité de la triangulation de Delaunay : autant que faire se peut, les triangles aplatis sont évités. La triangulation qui minimise le plus grand angle pourrait également être utilisée dans ce but, mais elle est beaucoup plus difficile à calculer [27].

Optimalité locale

La démonstration de la propriété précédente peut être fait en utilisant un critère dit «d'optimalité locale».

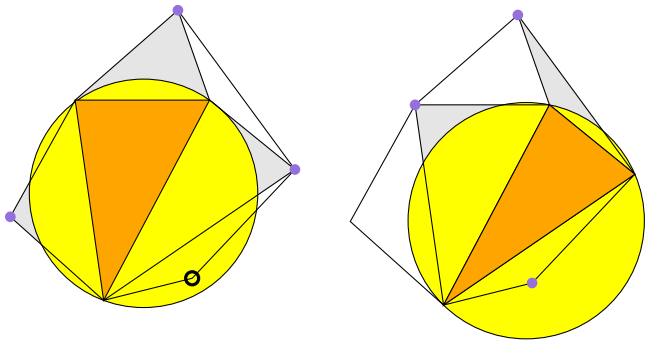

FIG. 3.3 – À gauche, le triangle colorié est localement de Delaunay mais pas globalement. À droite, le triangle n'est pas de Delaunay, même localement.

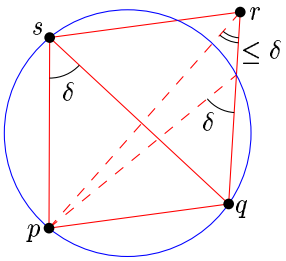

FIG. 3.4 – Théorème de l'arc capable

Une triangulation est dite localement de Delaunay si le cercle circonscrit à chaque triangle ne contient pas les sommets des triangles voisins (figure 3.3).

Théorème 3.4 *Une triangulation localement de Delaunay est* la *triangulation de Delaunay.*

Démonstration Si il existe un triangle f qui contient un point p dans une triangulation localement de Delaunay, on montre que l'on peut atteindre p en utilisant les relations de voisinage dans la triangulation et en utilisant uniquement des triangles dont le cercle circonscrit contient p pour aboutir à une contradiction. ■

Lemme 3.5 *Étant donné 4 points en position convexe, la triangulation de Delaunay est maximale pour l'ordre lexicographique des angles.*

Démonstration Il s'agit du thérème 3.3 pour 4 points seulement. Cette propriété résulte directement du théorème de l'arc capable qui indique que l'ensemble des points «voyant» un segment donné ous un angle donné est un arc de cercle. Si *pqrs* est un quadrilatère convexe, on en déduit que l'angle

prq est plus petit que l'angle $psq = \delta$ si et seulement si r est à l'extérieur du cercle passant par p, q et s (voir figure 3.4). ■

Démonstration (du théorème 3.3). Considérons la triangulation maximale pour l'ordre lexicographique des angles. D'après le lemme 3.5, une arête de cette triangulation est localement de Delaunay, sinon le flip de cette arête produirait une triangulation en augmentant le plus petit angle (parmi les angles modifiés). La triangulation maximisant le plus petit angle est donc localement de Delaunay, c'est donc la triangulation de Delaunay d'après le théorème 3.4. ■

3.1.4 Applications théoriques

La triangulation de Delaunay permet de répondre à un certain nombres de questions géométriques simples.

Recherche de plus proche voisin

Étant donné $\mathcal{S}$ un ensemble de points du plan, et $p \in \mathcal{S}$ quel est le plus proche voisin q de p dans $\mathcal{S}$?

C'est un point q tel que pq soit une arête de $\mathcal{DT}(\mathcal{S})$.

En effet, si q est le plus proche voisin de p, le cercle de centre p passant par q est vide, et donc a fortiori, le cercle de diamètre pq est vide et pq est une arête de Delaunay.

Plus grand cercle vide

Étant donné $\mathcal{S}$ un ensemble de points du plan, quel est le plus grand cercle ne contenant aucun point de $\mathcal{S}$ centré à l'intérieur de l'enveloppe convexe de $\mathcal{S}$?

C'est un cercle circonscrit à un triangle de Delaunay.

Un cercle vide passant par moins de 3 points peut toujours être agrandi en déplaçant un peu son centre. Les cercles passant par 3 points correspondent aux triangles de Delaunay.

Paires de points à distance bornée

Étant donné $\mathcal{S}$ un ensemble de points du plan, quelles sont les paires de points $(p, q) \in \mathcal{S}^2$ telle que $|pq| \leq \delta$?

Il suffit à partir de chaque p d'explorer le graphe de $\mathcal{DT}(\mathcal{S})$ en bornant la recherche à une distance δ de p [23, 52].

Arbre couvrant minimal

Étant donné $\mathcal{S}$ un ensemble de points du plan, on recherche l'arbre couvrant de longueur minimale $\mathcal{MST}(\mathcal{S})$ (*minimum spanning tree*). Un arbre couvrant est un graphe dont les sommets sont les points de $\mathcal{S}$, qui soit connexe et sans cycle, sa longueur est la somme des longueurs de toutes ses arêtes.

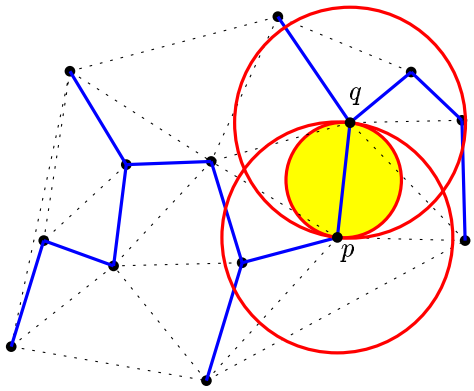

FIG. 3.5 – Arbre couvrant minimal

Théorème 3.6 *L'arbre couvrant de longueur minimale est inclus dans la triangulation de Delaunay.*

Démonstration Soit pq une arête de $\mathcal{MST}(\mathcal{S})$, on va montrer que pq est une arête de Delaunay. Si on enlève pq, $\mathcal{MST}(\mathcal{S})$ est déconnecté en deux parties, on va appeller $\mathcal{S}_p$ et $\mathcal{S}_q$ la partition de $\mathcal{S}$ ainsi obtenue. Le cercle de centre p et passant par q ne contient pas de point de $\mathcal{S}_q$, sinon en reliant p à ce point on obtiendrait un arbre couvrant de longueur inférieure. De même, le cercle de centre q et passant par p ne contient pas de point de $\mathcal{S}_p$. Le cercle de diamètre pq est inclus dans l'intersection de ces deux cercles, il est donc vide et pq est une arête de Delaunay (figure 3.5). ■

3.2 Domaines d'applications

La triangulation de Delaunay a été très largement étudiée car elle a prouvé son utilité dans de nombreux domaines. Nous en énumérons quelques uns ci dessous.

- Maillages et éléments finis.
  Pour les calculs par éléments finis, on cherche à approximer une fonction sur un certain domaine de l'espace. Pour cela, ce domaine est divisé (maillé) en éléments simples tels que tétrhèdres ou hexaèdres, et sur chaqun de ces éléments on approxime la fonction par interpolation des valeurs aux sommetslesquels on a recours à une interpolation des valeurs de la fonction aux sommets du maillage. Pour les maillages dits *non structurés* on a souvent recours à la triangulation (tétrahédrisation) de Delaunay pour ses bonnes propriétés angulaires (théorème 3.3) qui garantisse l'absence de triangles aplatis [30, 5].
- Planification de trajectoire (Voronoï). Si on appelle les sites de $\mathcal{S}$ des obstacles, le diagramme de Voronoï permet alors de connaître l'obstacle

le plus proche d'une position donnée, et en particulier de savoir si cet obstacle fait ou non obstacle au placement d'un robot circulaire centré sur cette position. On peut ainsi utiliser le diagramme de Voronoï pour planifier le mouvement d'un robot circulaire avec des obstacles ponctuels. Des généralisations du diagramme de Voronoï permettent d'obtenir des algorithmes pour planifier le mouvement d'un robot polygonal dans un environnement polygonal [36].

- Reconstruction tri-dimensionnelle.
  Dans de nombreuses applications on se trouve confronté au problème suivant : étant donné un ensemble de points que l'on sait appartenir au bord d'un objet on cherche à relier les points entre eux pour retrouver la forme de cette objet sous forme d'une surface triangulée. Delaunay permettant de relier les points entre eux par proximité, il semble naturel de chercher cette surface dans la tétrahédrisation. Nous donnons plus de détails sur cette application à la reconstruction au chapitre 13. En pratique, l'objet à reconstruire et l'ensemble de points que l'on a numérisé sur sa surface peuvent avoir diverses provenance, il peut par example s'agir d'un organe numérisé à l'aide d'une technique d'imagerie médicale, d'une maquette numérisé avec un capteur laser, d'une couche géologique sur laquelle on a certaines données sismiques...
- Modèles de terrains, reconstruction de surfaces. Un cas particulier de reconstruction tridimensionnelle est celui des terrains. Les données sont un ensemble de points mesurés sur un terrain, par exemple à partir d'images satellites, et l'on cherche à modéliser la surface de ce terrain. Une technique très fréquente consisite alors à utiliser la triangulation de Delaunay bi-dimensionnelle de la projection horizontale de ces points.

3.3 Représentation de la triangulation de Delaunay

Si on veut utiliser ou calculer la triangulation de Delaunay, il est bien entendu nécessaire d'utiliser une structure de données pour représenter celle ci.

En fait plusieurs structures sont envisageables. Il est possible d'utiliser des structures générales de cartes planaires mais nous préférerons une représentation spécifique des triangulations dans laquelle l'objet principal est le triangle.

Plus précisément nous maintiendrons en mémoire deux types d'objets : les triangles et les sommets. Un triangle connaît ses trois voisins et ses trois sommets. Un sommet connaît ses coordonnées et un triangle qui lui est incident. Une arête est représenté implicitement comme une certaine arête d'un triangle donné.

Il est également pratique de représenter une arête de l'enveloppe convexe pq comme un triangle $pq\infty$ en ajoutant un sommet virtuel «à l'infini». L'ajout d'un tel sommet permet à la triangulation complête de ne pas avoir de bord, et d'être topologiquement équivalente à la triangulation d'une sphère [6].

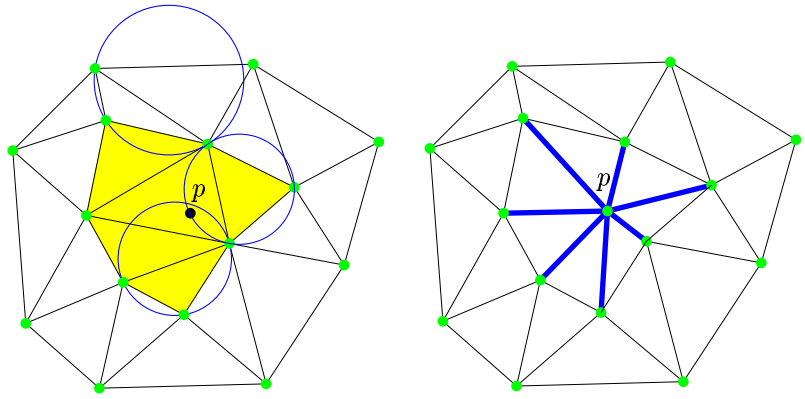

FIG. 3.6 – À gauche $\mathcal{DT}(\mathcal{S})$, les triangles coloriés doivent être supprimés. À droite $\mathcal{DT}(\mathcal{S} \cup \{p\})$.

3.4 Premiers algorithmes

3.4.1 Algorithme incrémental

Une première méthode de calcul de la triangulation de Delaunay consiste à ajouter les points un par un en mettant à jour la triangulation à chaque nouveau point.

En se reportant au théorème 3.1, on sait qu'un triangle est de Delaunay si et seulement si son cercle circonscrit est «vide». Les triangles de $\mathcal{DT}(\mathcal{S})$ dont le cercle circonscrit ne contient pas p seront toujours vide si on considère l'ensemble $\mathcal{S} \cup \{p\}$ et font donc partie de $\mathcal{DT}(\mathcal{S} \cup \{p\})$. De même un triangle de $\mathcal{DT}(\mathcal{S} \cup \{p\})$ n'ayant pas p comme sommet faisait déjà partie de $\mathcal{DT}(\mathcal{S})$ puisque son cercle circonscrit ne contient pas de point de $\mathcal{S} \cup \{p\}$ donc a fortiori pas de point de $\mathcal{S}$.

La triangulation $\mathcal{DT}(\mathcal{S} \cup \{p\})$ se déduit donc de $\mathcal{DT}(\mathcal{S})$ en supprimant tous les triangles dont le cercle circonscrit contient p et en retriangulant cette zone depuis p comme indiqué sur la figure 3.6 (on a comme corollaire que la zone détruite est étoilée par rapport à p).

Complexité

La complexité d'un algorithme est le nombre d'opération élémentaires nécessaire pour mener à bien une opération plus complexe.

Ici l'insertion de p dans $\mathcal{DT}(\mathcal{S})$ nécessite de tester l'inclusion de p dans les cercles circonscrits aux différents triangles, puis de trouver les nouveaux triangles à construire.

Si la recherche des triangles à détruire est faite en testant le nouveau point par rapport au cercle circonscrit à chaque triangle de la triangulation, il faudra donc $O(n)$ tests de cocircularité pour déterminer ces triangles, la construction

des nouveaux triangles est également linéaire et on aboutit à un algorithme quadratique pour construire la triangulation de n points.

3.4.2 Algorithme incrémental avec marche

Si les points sont bien réparti dans le plan (par exemple une distribution uniforme dans un carré ou des points suivant une loi de Poisson) on peut accélérer la recherche des triangles [38]. On remarque que l'ensemble des triangles à détruire est connexe et que le triangle qui contient le nouveau point en fait partie.

On peut donc chercher le triangle qui contient p, puis ne faire des tests de cocircularité que pour des triangles adjacents à des triangles dont le cercle à déjà été reconnu comme contenant p. Si k est le nombre de triangles détruits, le nombre de tests de cocircularité effectués est alors $k-1$ avec réponse positive (le triangle contenant le point n'a pas besoin d'être testé) et $k+2$ avec réponse négative ; le degré de p après son insertion étant $k+2$. Pour des points répartis aléatoirement, la valeur moyenne de k est 4, on fait donc en moyenne 9 tests de cocircularité.

Une «marche» dans la triangulation permet de trouver le triangle contenant p (figure 3.7). Si on connaît le triangle contenant un point ω alors on se propage dans la triangulation en utilisant les liens de voisinage entre les triangles et en visitant tous les triangles traversés par le segment ωp. Si les triangles sont bien réparti, on visite ainsi $O(\sqrt{n})$ triangles [10]. Une telle marche n'utilise que des tests d'orientation de trois points.

La complexité totale d'un algorithme incrémental de construction de Delaunay basé sur une telle localisation est donc de $O(n\sqrt{n})$ si les points sont répartis aléatoirement dans le plan. Plus précisément on utilisera $O(n)$ tests de cocircularité et $O(n\sqrt{n})$ tests d'orientation de trois points.

3.4.3 Algorithme par bascule de diagonales.

Étant donné une triangulation $\mathcal{T}(\mathcal{S})$ des points de $\mathcal{S}$, on peut obtenir une autre triangulation $\mathcal{T}'(\mathcal{S})$ en *basculant une diagonale*. Cette opération de bascule (ou *flip*) consiste à choisir deux triangles pqr et rqs adjacents dans $\mathcal{T}(\mathcal{S})$ tel que le quadrilatère $pqsr$ soit convexe, on enlève alors l'arête qr de la triangulation et on ajoute l'arête ps (voir figure 3.8).

En utilisant suffisamment d'opérations de bascule, on peut ainsi passer de n'importe quelle triangulation de $\mathcal{S}$ à n'importe quelle autre [33, 34].

Étant donné une triangulation quelconque, on peut alors obtenir la triangulation de Delaunay par une suite de flip. En fait, si pqr et rqs sont deux triangles on va flipper l'arête qr si et seulement si le cercle passant par p, q et r contient s. Si l'on poursuit cette opération de flip jusqu'à ce que plus aucune arête de la triangulation ne corresponde au critère on obtient une triangulation localement de Delaunay et donc d'après le théorème 3.4 la triangulation de Delaunay.

On peut montrer par transformation de la triangulation de Delaunay en enveloppe convexe en dimension trois (théorème 4.3) qu'une arête «flippé» ne

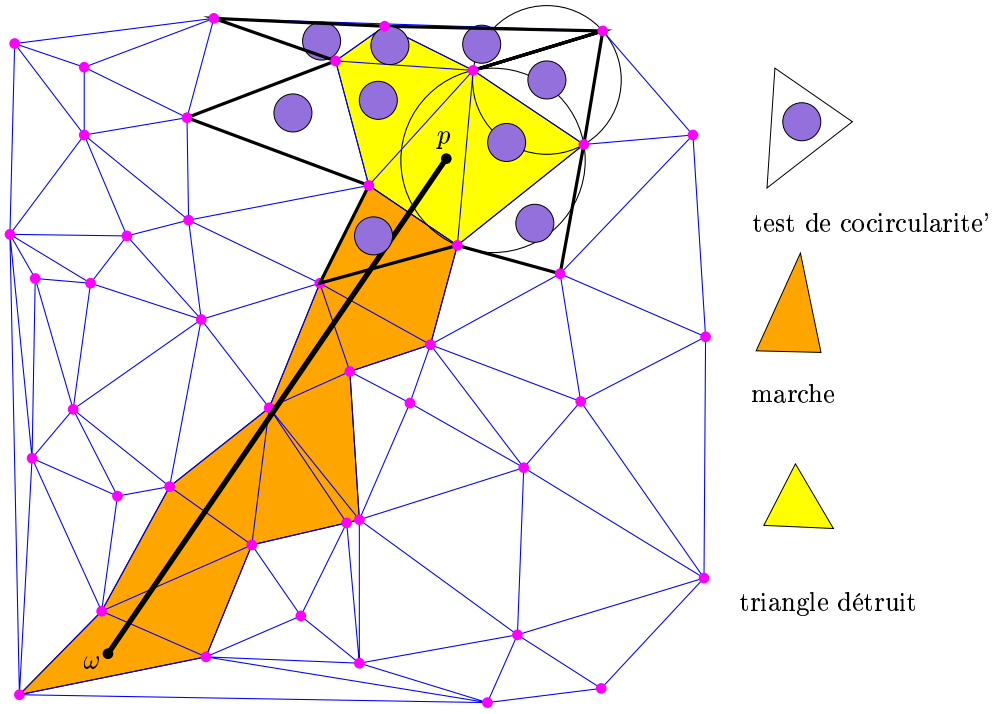

FIG. 3.7 – Localisation d'un point dans Delaunay par marche.

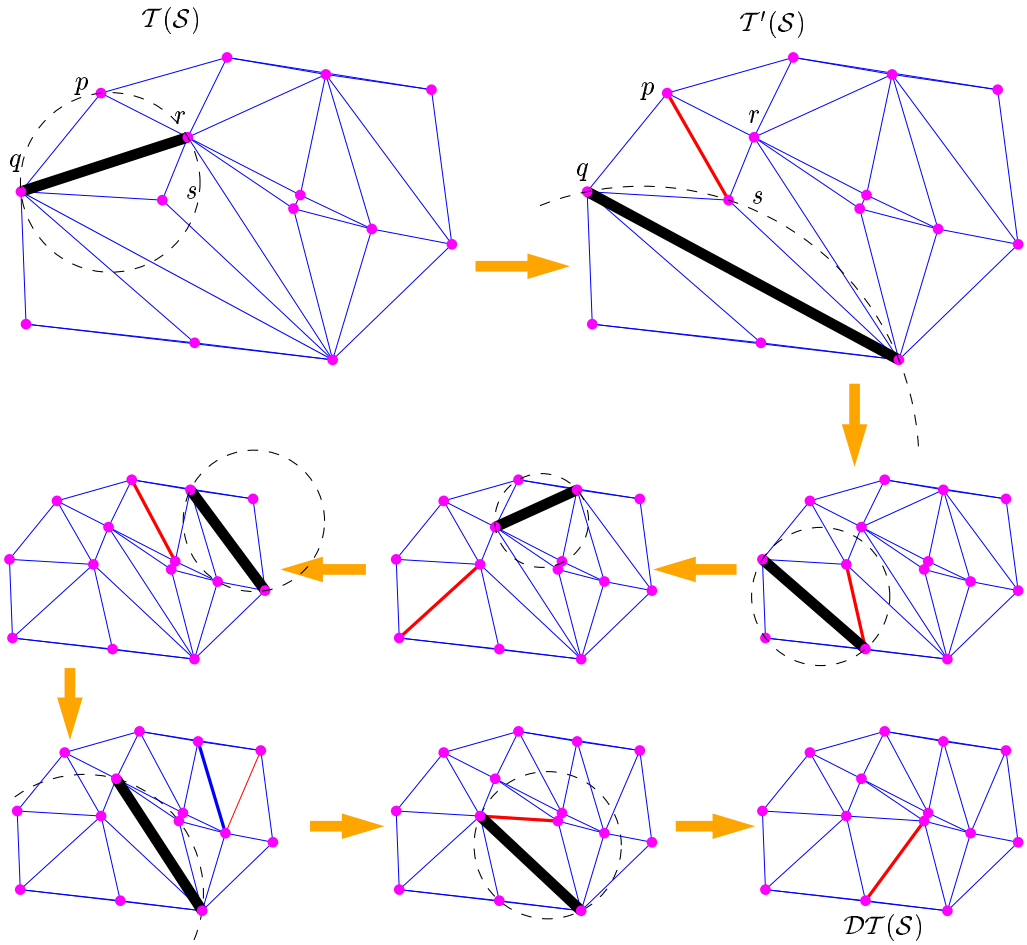

FIG. 3.8 – Bascule de diagonale. Bascule successive jusqu'à obtention de la triangulation de Delaunay.

peut jamais être reconstruite, comme il n'y a que $O(n^2)$ arêtes possibles on obtiens triviallement une complexité quadratique. Pour certains exemples, un nombre quadratique de flip sera effectivement nécessaire [34].

Cette propriété d'unicité de l'optimum local et donc de convergence de l'algorithme de bascule est propre à Delaunay et n'est pas vérifiée pour d'autres triangulations, telle que la triangulation qui minimise le plus grand angle ou la triangulation de longueur totale minimale.

Chapitre 4

Les solutions de la géométrie algorithmique

4.1 Borne inférieure

Pour un problème tel que le calcul de la triangulation de Delaunay, on cherche à établir des bornes inférieures sur la complexité d'un algorithme permettant de résoudre le problème. Une telle borne est alors noté $\Omega(f(n))$ ce qui signifie qu'il existe une constante β telle que pour n assez grand on ne peut pas trouver d'algorithme résolvant tous les problèmes de taille n an moins de $\beta f(n)$ opérations. Il faut bien sur préciser ce que l'on apelle opérations, c'est à dire préciser le modèle de calcul. Le modèle le plus fréquent est le *information theory bound* dans lequel un test ne peut avoir que deux réponses, on compte alors le nombre de branchement binaire nécessaire à l'algorithme. On peut également envisager des modèles plus puissants dans lesquels on peut avoir des branchements à sortie multiples (accés aléatoire à la mémoire par exemple) ou des modèles moins puissants où on restreint les opérations autorisées [28].

Il y a trois grandes techniques pour établir des bornes inférieures, la plus simple consiste tout simplement à regarder la taille du résultat, la seconde procède par comparaison avec un problème connu tel que le tri de n nombres, et finalement on peut montrer des bornes inférieures en regardant le nombre minimal de branchement nécessaire à l'algorithme pour distnguer les différentes réponses possibles.

Nous allons tout d'abord utiliser la dernière technique pour montrer la borne inférieure classique pour le problème du tri.

Théorème 4.1 *Le tri de n nombres nécessite $\Omega(n \log n)$ comparaisons.*

Démonstration Un algorithme de tri permet en fait de calculer la permutation de n nombres qui va permettre de les remettre dans l'ordre, deux tableaux de n nombres correspondant à des permutations différentes produiront nécessairement des executions différentes de l'algorithme de tri. On peut

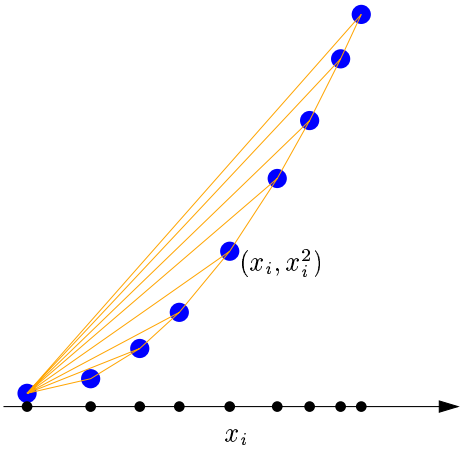

FIG. 4.1 – Triangulation de Delaunay de points sur la parabole.

considérer un arbre binaire correpondant à toutes les executions possibles de l'algorithme, un nœud de cet arbre correspond à une décision (comparaison) et le chemin entre deux nœuds à une séquence de calcul sans tests, une feuille de cette arbre correspond à la terminaison d'une execution de l'algorithme. D'après ce qui précéde, un tel arbre doit avoir au moins autant de feuille que de permutation de n nombres, c'est à dire $n!$, par conséquent sa hauteur (longueur de la plus grande branche de la racine à une feuille) est au moins $\log_2 n! \simeq n \log n$. Clairement, la longueur du chemin de la racine à une feuille est le nombre de tests effectués par l'éxecution correspondante de l'algorithme de tri et donc une borne inférieure pour le temps d'exécution de l'algorithme. ■

Nous allons maintenant prouver une borne inférieure pour le calcul de la triangulation de Delaunay. Cette borne est établie en montrant que la triangulation de Delaunay est un problème «plus dur» que le tri.

Théorème 4.2 *Le calcul de la triangulation de Delaunay de n points du plan nécessite $\Omega(n \log n)$ opérations*

Démonstration Étant donné un ensemble de n nombres positifs $\{x_1, x_2, \ldots x_n\}$ (non ordonné) on peut construire l'ensemble de points $\{(x_1, x_1^2), (x_2, x_2^2), \ldots (x_n, x_n^2)\}$ (voir figure 4.1). Supposons que l'on dispose d'un algorithme permettant de calculer la triangulation de Delaunay de ces points en temps $f(n)$, il est alors facile de récupérer en temps $O(n)$ l'ordre des points en x et donc de trier l'ensemble $\{x_1, x_2, \ldots x_n\}$. On a donc déduit de notre algorithme pour Delaunay un algorithme de tri de complexité $f(n) + O(n)$, comme on sait par le théorème 4.1 qu'un tel algorithme a une complexité minimale $\Omega(n \log n)$, on en déduit que $f(n) \geq \Omega(n \log n)$. ■

Maintenant que nous savons que le problème ne peut pas être résolu en moins de $n \log n$ opérations, nous allons proposer quelques algorithmes ayant cette complexité. On peut remarquer que cette borne inférieure s'applique au problème brut du calcul de la triangulation de Delaunay de n points sans que l'on connaisse d'autres information sur ces points que leurs coordonnées. Dans un certain nombre de cas particulier on peut obtenir des algorithmes plus rapides. Par exemple si les points sont les sommets d'un polygone convexe connu, alors la triangulation peut être obtenue en temps linéaire, soit par un algorithme déterministe [1] soit par un algorithme randomisé très simple (voir paragraphe 4.5 [13]). Si on connait un sous-ensemble du résultat, alors on peut également obtenir un algorithme plus rapide par exemple si l'on connait l'arbre couvrant de longueur minimale [16]. Par contre, connaître l'ordre des points en abcisse ne permet pas de battre la borne de $\Omega(n \log n)$ [24] contrairement au cas d'une triangulation quelconque de l'ensemble des points [11, 49]. La connaissance de la triangulation d'un sur-ensemble des points permet également d'atteindre un algorithme linéaire [12].

4.2 Division fusion

La technique de division fusion (*divide and conquer*) est très classique dans de nombreux algorithmes, le problème est divisé en deux sous problèmes de taille raisonablement équilibrés en temps linéaire, chaqu'un des deux sous-problèmes est résolu récursivement et enfin les résultats doivent être fusionné en temps linéaire. On obtient ainsi une complexité globale de $O(n \log n)$ pour résoudre un problème de taille n.

Nous allons maintenant passer à la description d'une solution division fusion pour la triangulation de Delaunay.

4.2.1 Division

La division de l'ensemble de points est effectuée en deux sous-ensemble de taille $\lfloor \frac{n}{2} \rfloor$ et $\lceil \frac{n}{2} \rceil$ autour de l'abcisse médiane. Cette division peut être faite en temps linéaire avec un algorithme de recherche de médian en temps linéaire déterministe [47] ou randomisé (paragraphe 4.5.1), mais il est plus judicieux de trier les points en abcisse dans un premier temps (en temps $O(n \log n)$, on accède ainsi facilement au médian de n'importe quelle sous-liste de la liste des points triés en x.

4.2.2 Fusion

On a donc une droite verticale séparant un ensemble de points bleu à gauche et un ensemble de points rouge à droite, on suppose les triangulations des points bleus et des points rouges calculées et on cherche à déduire la triangulation de l'ensemble des points (figure 4.2).

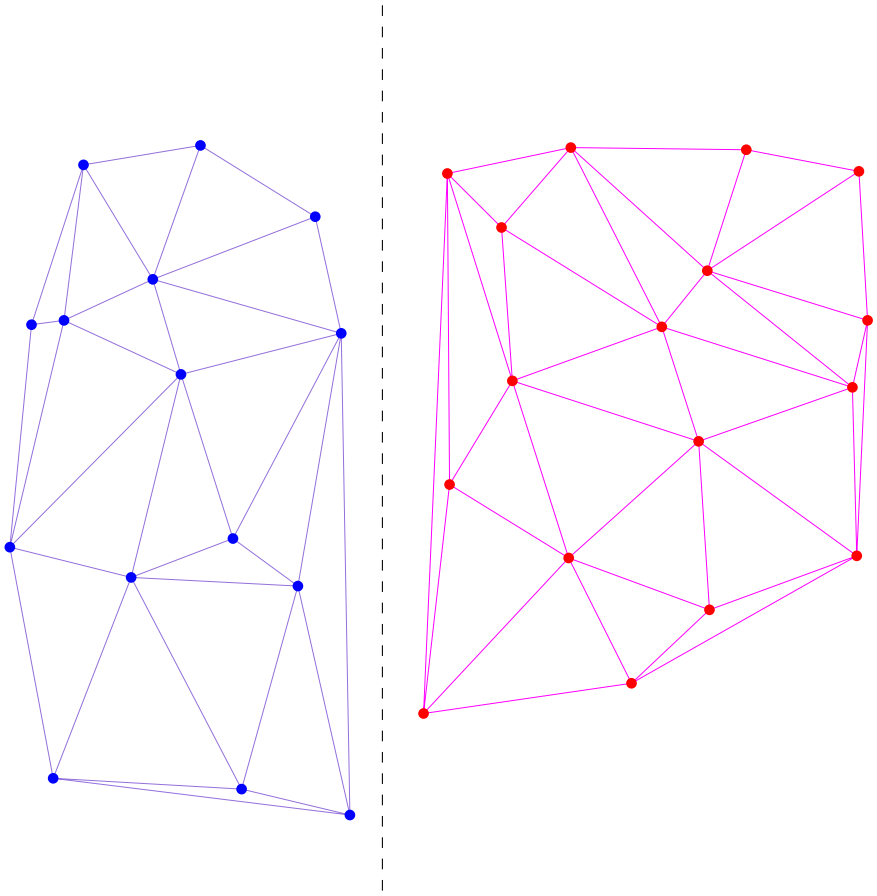

FIG. 4.2 – Division-fusion. Étape de division.

C'est à dire que l'on doit éliminer un certain nombre de triangles bleus contenant des points rouges et vice-versa, et relier ce qui reste des deux triangulations par des triangles bicolores. On va alors procéder de haut en bas. Sur la figure 4.3 on a déjà crée trois triangles bicolores dont le triangle prt et on cherche maintenant à créer un triangle en dessous de l'arête pr. On va tout d'abord chercher le cercle passant par p, r et un point rouge et ne contenant ni points rouges ni t, on examine pour cela les triangles incidents à r dans la triangulation rouge en tournant dans le sens direct autour de r. Le cercle circonscrit au premier triangle rs_1s_2 contient p, on en déduit que l'arête rs_1 doit donc être éliminé de la triangulation finale. Il en va de même pour les arêtes rs_2 et rs_3. Le cercle passant par rs_4s_5 ne contiens pas p, et donc de manière équivalente le cercle circonscrit à prs_4 ne contient pas de points rouge car il est inclus dans les cercles passant par rs_3s_4 et rs_4s_5. Si le point recherché pour former une triangle avec l'arête pr est rouge, ce sera donc s_4. De même, le cercle passant par pq_1q_2 contient r, l'arête pq_1 est donc supprimé, le cercle pq_2q_3 ne contient pas r, le second triangle candidat est donc prq_2 qui ne contient pas de points bleus. Il ne reste qu'à comparer prs_4 avec prq_2, s_4 étant dans le cercle passant par prq_2, c'est donc prs_4 le nouveau triangle de Delaunay (voir figure 4.4). Il ne reste qu'à itérer cette recherche pour trouver maintenant le triangle en dessous de l'arête ps_4, il est à noter que pour cette recherche le voisin q_1 de p n'est pas réexaminé car l'arête pq_1 a été enlevé lors de la recherche du triangle prs_4 (figure 4.5). On aboutit finallement à la triangulation finale de la figure 4.6. La complexité de cette phase de fusion est clairement linéaire puisque à chaque fois qu'un point est examiné puis rejeté, on détruit une arête de l'une de la triangulation rouge ou de la triangulation bleue.

Remarques

L'algorithme que nous avons décrit ci dessus basé sur une séparation des points en abcisse est classiquement décrit pour le problème du diagramme de Voronoï et nous avons ici transposé l'algorithme pour l'exprimer directement en termes de triangulation de Delaunay.

On a obtenu une complexité de $O(n \log n)$ ce qui est ce que l'on peut espérer de mieux dans le cas le pire, toutefois si les points sont bien réparti dans le plan, on a interêt à alterner des subdivisions verticales et horizontales. La phase de fusion est alors identique, seule la phase de subdivision est modifiée. Le tri préliminaire des points en abcisse pour trouver facilement les médians est alors infructueux, on peut utiliser un algorithme de médian en temps linéaire déterministe ou randomisé (voir au paragraphe 4.5), ou si les hypothèses sur la répartition des points le permettent calculer une droite séparant l'ensemble de points de manière suffisamment équilibrée avec une bonne probabilité [51, 39].

On peut envisager d'autres modes de division, la phase de fusion est alors plus complexe.

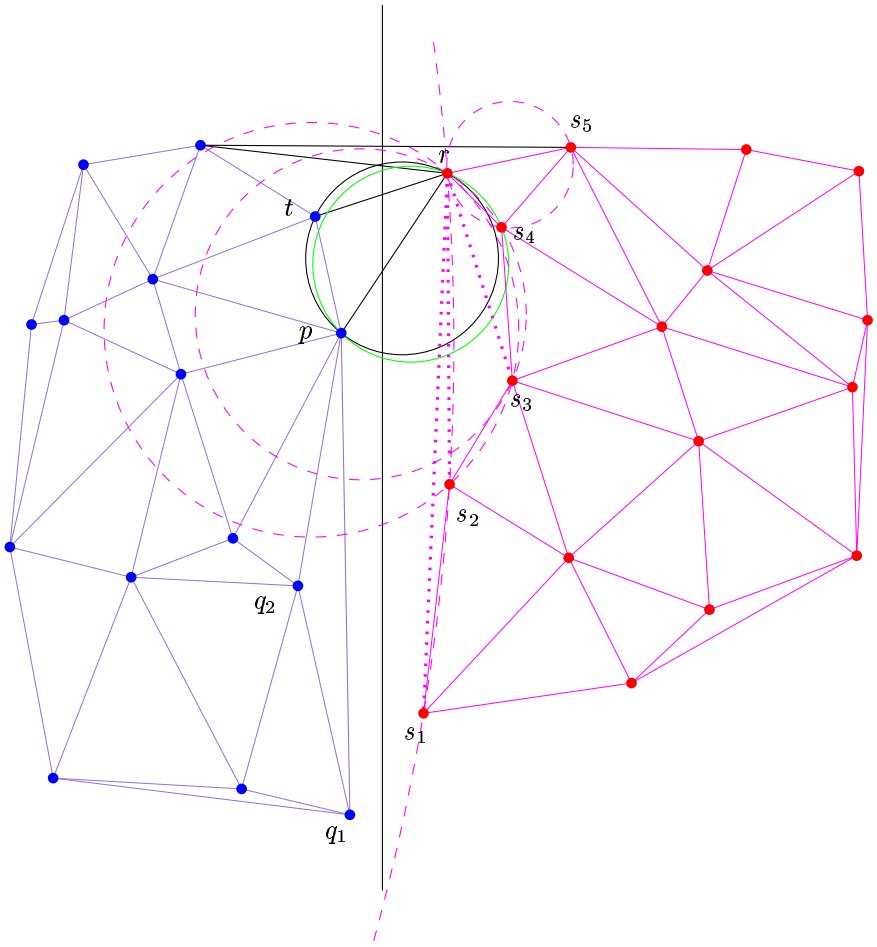

FIG. 4.3 – Division-fusion. Étape de fusion, recherche du triangle prs_i dont le cercle ne contiens pas de points rouges.

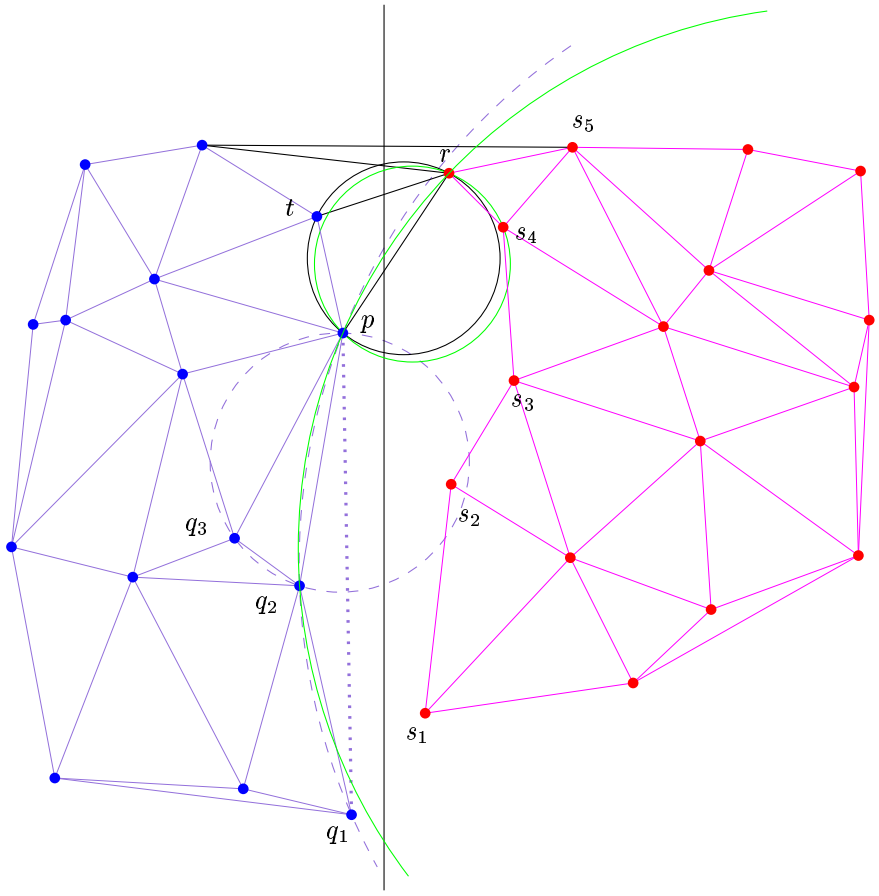

FIG. 4.4 – Division-fusion. Étape de fusion, recherche de prq_j et comparaison entre q_j et s_i.

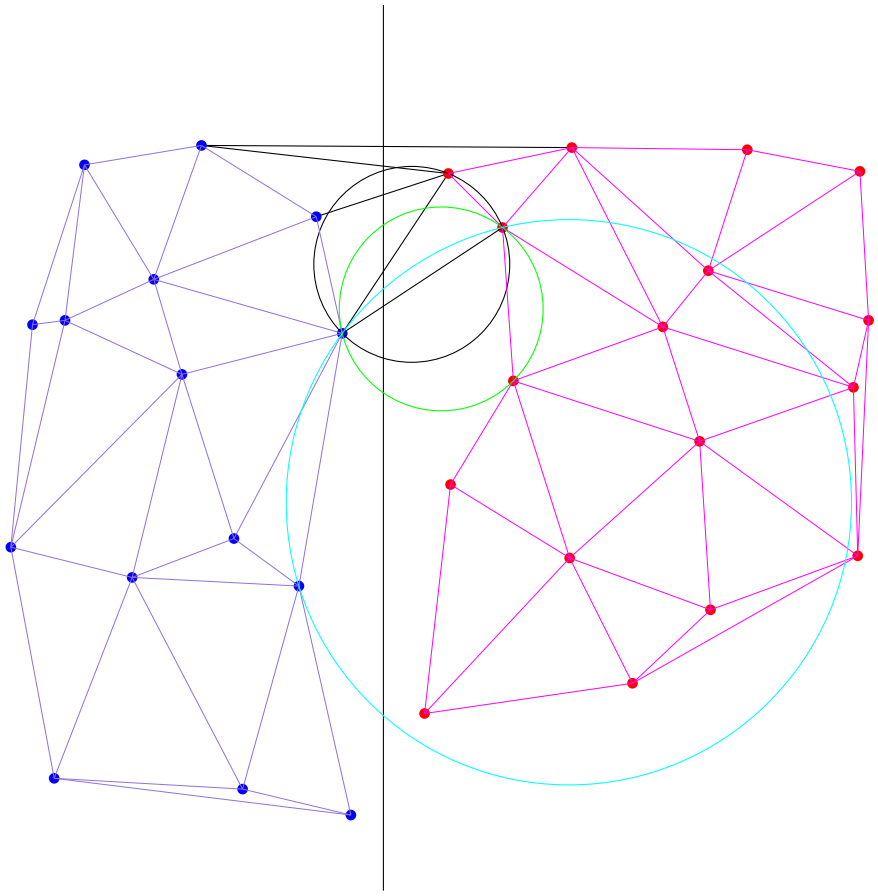

FIG. 4.5 – Division-fusion. Étape de fusion, recherche du triangle suivant.

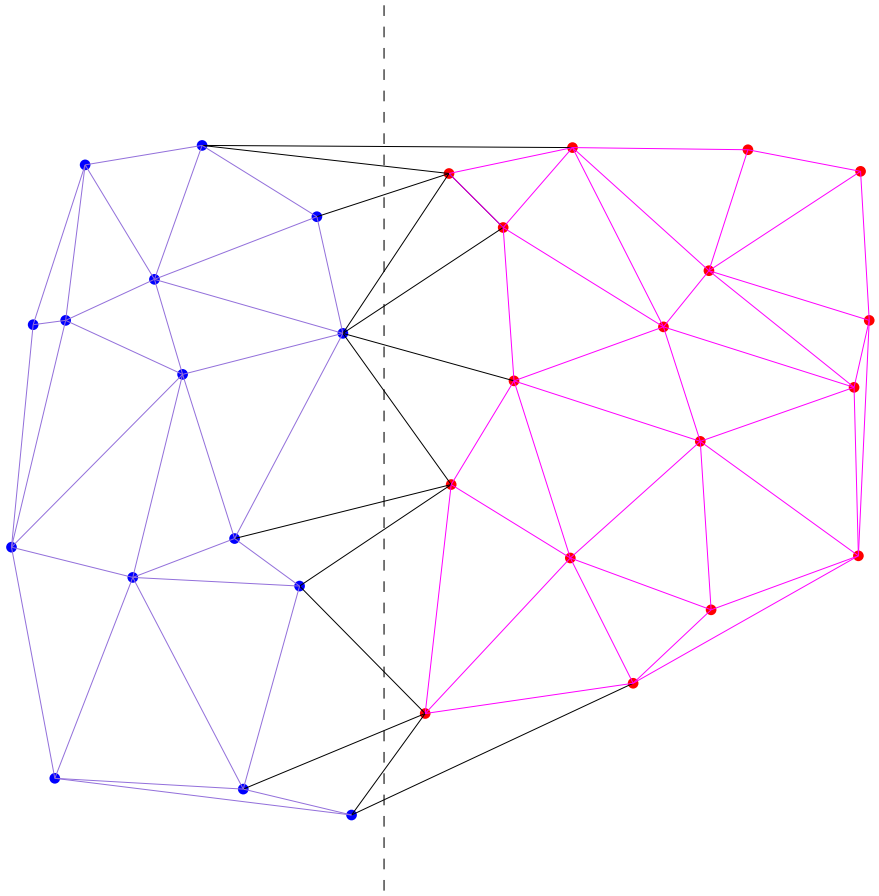

FIG. 4.6 – Division-fusion. Étape de fusion, résultat final.

4.3 Balayage

Après l'approche division-fusion, qi est loin d'être propre à la géométrie, nous allons voir une technique plus spécifique : le balayage. Les deux algorithmes de balayage les plus célébres sont celui de Bentley-Ottmann [4] pour la calcul des intersections dans un ensemble de segments du plan, et celui de Fortune [29, 32] pour le calcul du diagramme de Voronoï. Nous allons présenter ci dessous une transposition de l'algorithme de Fortune pour l'exprimer directement en termes de triangulation de Delaunay.

Le principe général d'un algorithme de balayage consiste à balayer le plan avec une droite verticale, au fur et à mesure du déplacement de cette droite, on découvre la structure recherchée. Comme la structure recherchée n'est modifiée que lorsque la droite passe dans certaines positions critiques, que l'on appellera événements, il faut être capable de trouver le prochain événement.

Pour la triangulation de Delaunay, la droite verticale qui balaye le plan se déplace de gauche à droite. Pour une position donnée de la droite de balayage tous les triangles dont le cercle circonscrit est vide et est à gauche de la droite de balayage sont certainement de Delaunay. On va donc supposer que ces triangles ont étés créés (figure 4.7). Maintenant quand la droite de balayage se déplace vers la gauche on va «fermer» des triangles, si qr et rs sont deux arêtes du bord de la région déjà triangulée alors le triangle qrs devra être formé lorsque la droite de balayage aura découvert son cercle circonscrit et que l'on est certain que ce cercle restera vide. Une position de la droite de balayage tangente à gauche d'un cercle est appellé événement cercle. Quand la droite de balayage se déplace elle peut également découvrir un nouveau point, une telle position de la droite de balayage est appellée événement point.

4.3.1 Structures de données

L'algorithme doit manipuler plusieurs structures de données, tout d'abord bien évidemment une structure pour représenter la triangulation de Delaunay comme décrite au paragraphe 3.3. Ensuite une structure pour gérer les événements à venir, cette structure est initialisée avec la liste des événements points et les événements cercles sont insérés et supprimés lors de l'algorithme. Une structure de type dictionnaire permettant des insertions et suppressions en temps logarithmique sera utilisée.

Enfin nous avons besoin d'une structure pour représenter le front de balayage (figure 4.8). Ce front de balayage est constitué par l'ensemble des segments du bord de la région déjà triangulée. Pour chaqu'un de ces segments, il existe un cercle vide passant par ces deux extrémités et tangent à la droite de balayage, l'ordre vertical des points de tangence de ces segments correspond à l'ordre des segments sur le bord de la région construite. Il faut faire attention qu'un segment peut apparaître deux fois dans la liste, «une fois de chaque coté».

Sur la figure 4.8 on a dessiné les cercles tangents à la droite de balayage (en violet) et les cercles circonscrits qui coupent la droite de balayage, le triangle correspondant n'est donc pas encore construit.

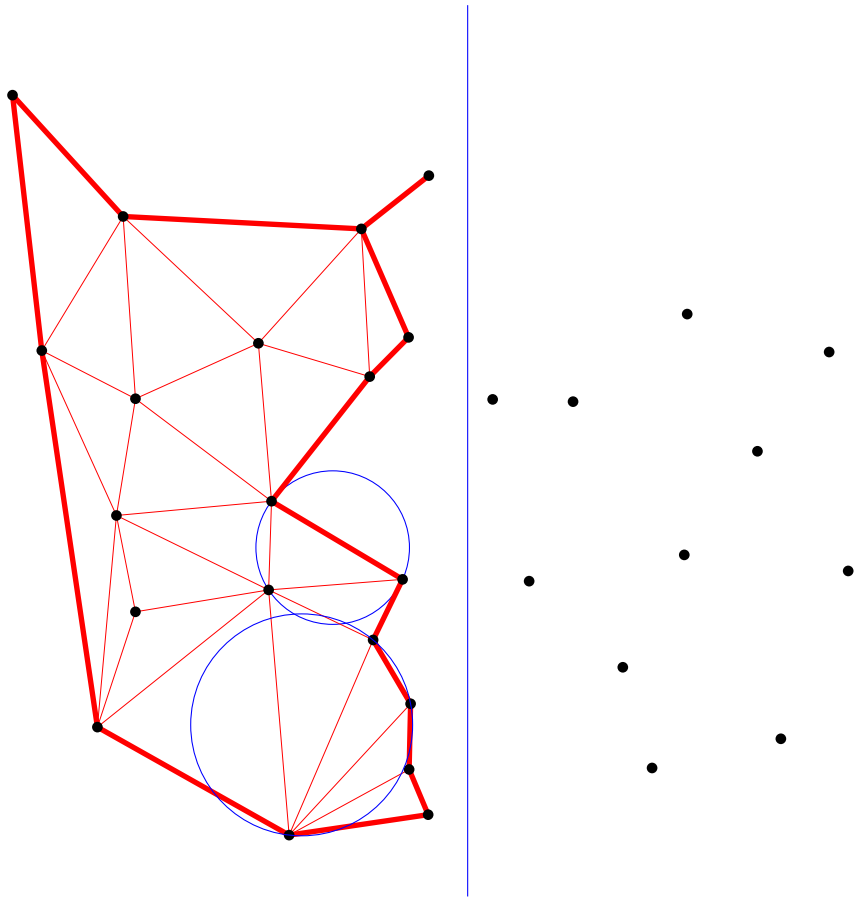

FIG. 4.7 – Algorithme de balayage.

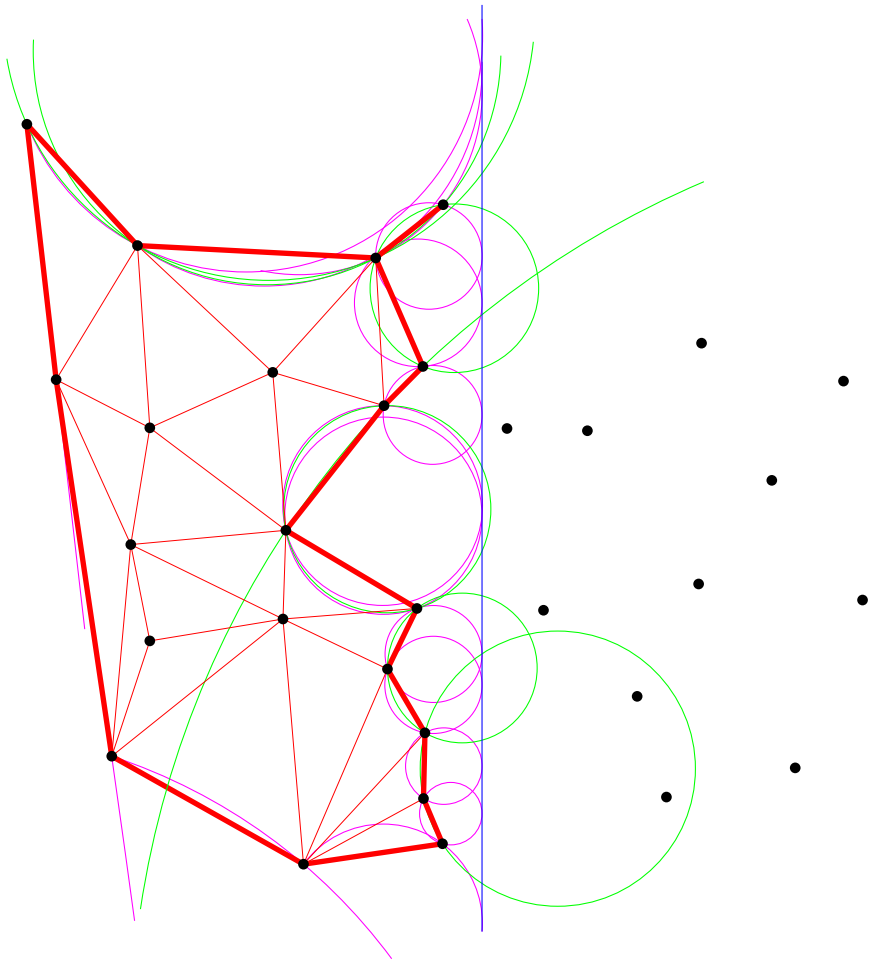

FIG. 4.8 – Algorithme de balayage, le front de balayage.

4.3.2 Événement cercle

Lorsque la droite de balayage devient tangente à un événement cercle, le triangle correspondant est alors certifié comme étant de Delaunay et peut donc être crée, il faut également modifier les structures auxiliaires c'est à dire la liste des événements et le front de balayage.

Si on appelle $pqrst$ la suite de point sur le pront de balayage et que l'on traite l'ébénement cercle pqr il faut (voir figure 4.9) :

— Créer le triangle pqr et le relier à ses voisins.

— Remplacer dans le front de balayage les arêtes qr et rs par l'arête qs.

— Enlever pqr de la liste des événements, puisque l'événement est traité.

— Enlever eventuellement les événements cercles pqr et rst puisqu'il ne correspondent plus à des segments du front de balayage (pqr sur la figure).

— Ajouter eventuellement les événements cercles pqs et qst (sur la figure qst n'est pas pris en compte car l'angle est convexe en s).

On remarque que l'orsque l'événement cercle se produit on est bien certain que le cercle est vide, alors que ce n'est pas le cas de tous les événements cercles, par exemple le cercle pqr n'est pas vide.

Le cercle passant par qrs est nécéssairement vide, en effet le triangle déjà construit incident à l'arête qr a un cercle vide, il ne contient donc pas s. On obtient ainsi que tout triangle construit est localement de Delaunay.

4.3.3 Événement point

Lorsque la droite de balayage découvre un nouveau point u, il faut raccorder celui ci à la partie déjà triangulée. Pour cela on va construire l'arête de Delaunay qui relie u à son plus proche voisin déjà découvert. Il suffit pour cela de localiser u dans l'ensemble des points de tangences des cercles associés aux arêtes du front de balayage (figure 4.8).

Si on appelle p ce point v le point qui précéde p et q le point qui suit p sur le front de balayage il faut (voir figure 4.10) :

— Localiser p dans le front de balayage.

— Insérer entre vp et pq les arêtes pu et up sur le front de balayage.

— Enlever l'événement p, puisque l'événement est traité.

— Enlever eventuellement l'événement cercle vpq puisqu'il ne correspond plus à un segment du front de balayage.

— Ajouter eventuellement les événements cercles vpu et upq.

4.3.4 Complexité

A chaque événement on découvre une nouvelle arête de la triangulation, il y a donc un nombre linéaire d'événements (paragraphe 3.1.3).

A chaque événement on effectue un nombre constant d'opérations sur le front de balayage et la liste des événements. Ces opérations d'insertion, de suppression ou de recherche sont toutes effectuées en temps logarithmique.

La complexité en temps de l'algorithme est donc $O(n \log n)$.

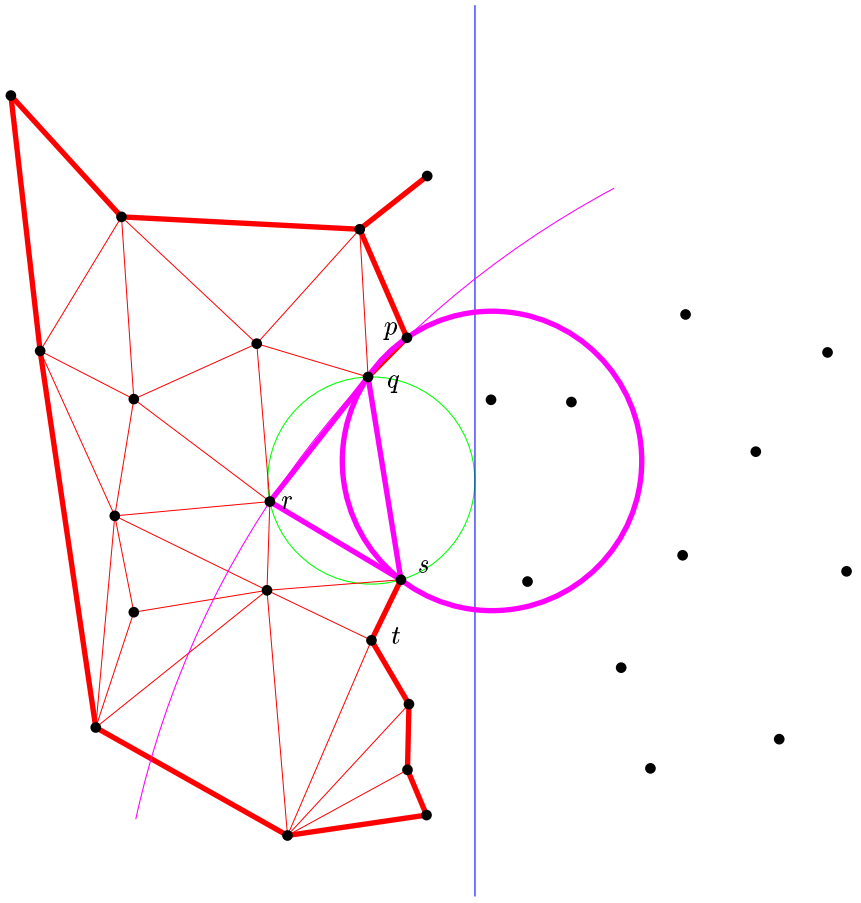

FIG. 4.9 – Algorithme de balayage, événement cercle.

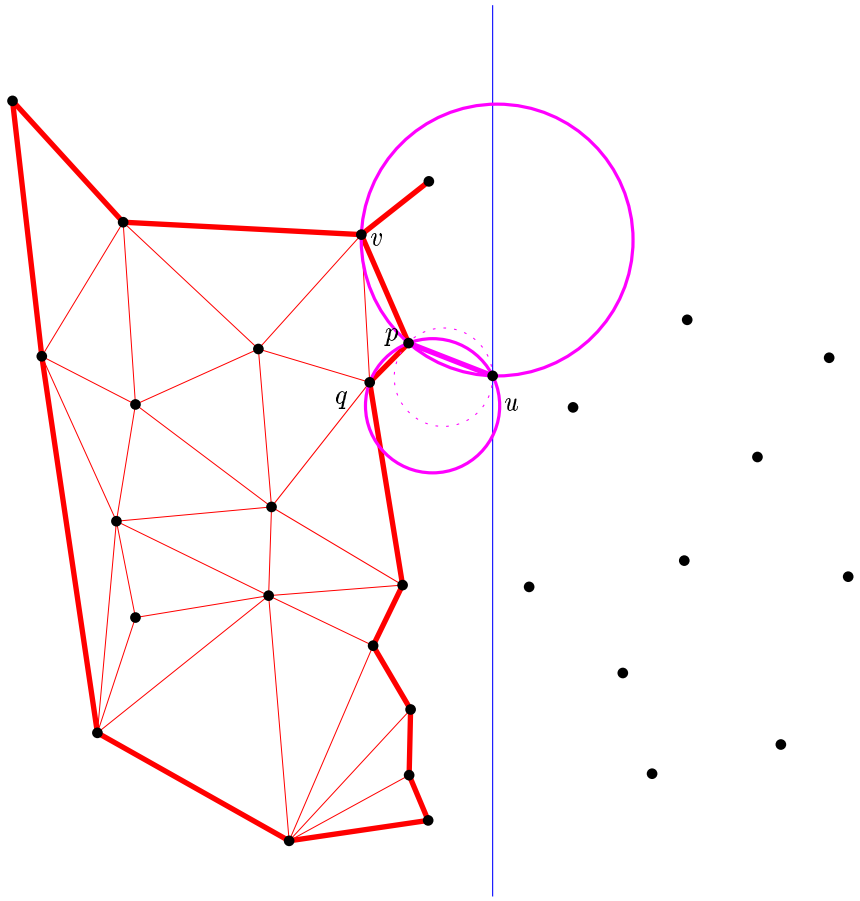

FIG. 4.10 – Algorithme de balayage, événement point.

Il ne reste plus qu'à controler la complexité en espace mémoire. La triangulation est bien entendu linéaire, le front de balayage est également linéaire, puisqu'il peut être vu comme une liste de points dans laquelle chaque point est répété au plus deux fois. Enfin la liste des événements a à tout instant une taille linéaire, çest évident pour les événements points, et pour les événements cercles cela découle directement du fait que chaque événement cercle correspond à une paire d'arête consécutive sur le front de balayage, et l'on viens de voir que celui ci avait une taille linéaire.

4.4 Transformation en enveloppe convexe

Un outil fréquent de la géométrie algorithmique consiste à utiliser un problème pour en résoudre un autre. On a déjà vu qu'un algorithme de calcul de la triangulation de Delaunay pouvait servir à trier des nombres à l'occasion du théorème 4.2. On va maintenant montrer que le calcul d'une enveloppe convexe en trois dimensions permet de trouver Delaunay en deux dimensions.

On va pour cela établir une correspondance entre des objets du plan et des objets de l'espace. A un point $p = (x_p, y_p)$ du plan, on associe sa projection $p^\star = (x_p, y_p, x_p^2 + y_p^2)$ sur le paraboloïde de révolution $\Pi : x^2 + y^2 - z = 0$. À un cercle C d'équation $x^2 + y^2 - 2ax - 2by + c = 0$ on associe le point $C^\star = (a, b, c)$ et le plan $C^\dagger : z - 2ax - 2by + c = 0$ conjugué de $C^\star$ par rapport à Π (voir figure 4.11).

On obtient alors facilement les propriétés

- p,q et r appartiennent au cercle C si et seulement si $p^\star$,$q^\star$ et $r^\star$ appartiennent au plan $C^\dagger$.
- s est à l'intérieur du cercle passant par p,q et r si et seulement si $s^\star$ est en dessous du plan passant par $p^\star$,$q^\star$ et $r^\star$.
- pqr est un triangle de Delaunay de $\mathcal{S}$ un ensemble de points si et seulement si $p^\star q^\star r^\star$ est une face de l'enveloppe convexe inférieure de $\mathcal{S}^\star = \{x^\star; x \in \mathcal{S}\}$.

Cette transformation entre sphères du plan etpoints d'un espace des sphères permets de nombreuses interprétations [2, 21, 3, 26]. Comme on viens de le voir, on peut donc utiliser un algorithme d'enveloppe convexe en trois dimensions pour résoudre Delaunay dans le plan. On peut de la même manière interpréter dans l'espace des sphères plusieurs types de requêtes et obtenir ainsi plusieurs résultats sur certaines généralisation de Delaunay ou Voronoï comme les diagrammes de puissance ou les diagrammes pour des métriques hyperboliques par exemple.

Théorème 4.3 *L'algorithme par flip du paragraphe 3.4.3 a une complexité quadratique.*

Démonstration L'interprétation de l'algorithme de flip en dimension 3 permet une preuve très facile à la fois de sa validité et de sa complexité. Une triangulation quelconque de l'ensemble de points, donne une fois relevée sur le parabloïde une surface triangulée. Imaginons que l'on regarde cette

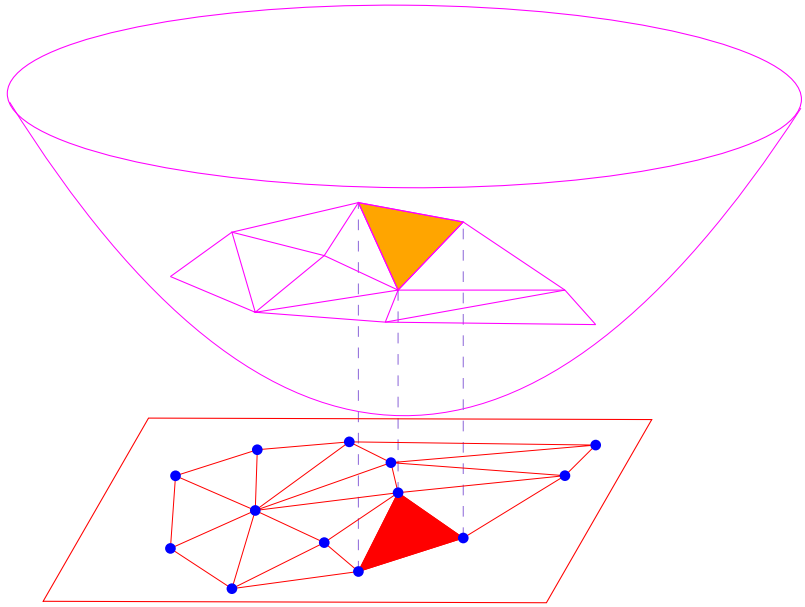

FIG. 4.11 – Triangulation de Delaunay et enveloppe convexe sur le paraboloïde.

surface par en dessous, un flip correspond alors à venir coller sur une arête concave de cette surface une tétraèdre pour venir remplir cette arête concave et la remplacer par une arête convexe. Chaque flip rajoute ainsi un tétrèdre jusqu'à ce que l'on ait complètement rempli l'enveloppe convexe des points, et que l'on ait atteint en projection la triangulation de Delaunay. En dimension 3, chaque arête qui disparaît est désormais cachée par un tétraèdre et ne peut en aucun cas être recrée. Comme il y a un nombre quadratique d'arêtes possibles on obtiens la complexité annoncée. ■

4.5 Algorithmes randomisés

Nous avons pour l'instant vu des algorithmes assez simples tel que le flip et l'algorithme incrémental (paragraphe 3.4) dont la complexité est quadratique, et d'autres algorithmes plus compliqué comme l'algorithme division-fusion ou le balayage permettant d'atteindre la complexité optimale $\Theta(n \log n)$. La recherche d'un moyen terme entre optimalité et simplicité de l'algorithme a conduit au développement d'une famille d'algorithmes dit *randomisés* [41, 44, 14, 7, 17]. Un algorithme n'est pas toujours complétement déterminé, il existe parfois plusieurs chemins menant au même résultat. Un algorithme randomisé est un algorithme qui doit choisir de nombreuses fois entre de tels chemins ce qui multiplie les possibilitées de déroulement de l'algorithme. L'analyse randomisé, au lieu de

regarder le chemin le plus long, c'est-à-dire le cas le pire va regarder la longueur moyenne du chemin. On obtiendra donc une complexité moyenne, la moyenne étant faite sur les «choix de chemins» effectués par l'algorithme. Cette notion de moyenne ne porte que sur l'exécution de l'algorithme et absolument pas sur les données, on ne fait aucune hypothèse sur l'ensemble de points. Les algorithmes randomisés permettent d'obtenir des algorithmes plus réalistes, plus simple, avec des dégradations de performances possibles mais peu probables.

Avant de traiter le cas de la triangulation de Delaunay, nous allons traiter un exemple très simple, celui de la recherche du médian.

4.5.1 Médian randomisé

Étant donné n nombres x_1, $x_2 \ldots x_n$ on cherche la valeur médiane x_i tel que $|\{x_j, x_j < x_i\}|, |\{x_j, x_j > x_i\}| \leq \frac{n}{2}$. Il existe pour ce problème un algorithme déterministe en temps linéaire sur le principe division fusion [47].

Une solution randomisée très simple pour trouver le nombre de rang p ($p = \frac{n}{2}$ pou le median) consiste à choisir un nombre x_k (qu'on appellera pivot) au hasard dans l'ensemble, à diviser l'ensemble en $\{x_j, x_j < x_k\}$ et $\{x_j, x_j > x_k\}$ de tailles respectives n_i et n_s, le nombre cherché est x_k si $n_i < p \leq n - n_s$ sinon on doit poursuivre la recherche récursivement, si $n_i \geq p$ on cherche le nombre de rang p dans $\{x_j, x_j < x_k\}$ ou celui de rang $p - n + n_s$ dans $\{x_j, x_j > x_k\}$ si $p > n - n_s$.

La complexité d'un tel algorithme peut être majorée assez facilement. À la première étape, tous les nombres sont comparés au premier pivot x_{k_1} soit $n - 1$ comparaisons. À l'issu de cette première étape notre ensemble de n nombres a été découpé en deux sous-ensembles de taille $i - 1$ et $n - i$ avec des probabilitées égales pour toutes les valeurs de i, la taille moyenne du plus gros des deux sous-ensemble est donc $\frac{1}{n}\left(\sum_{i=1}^{\frac{n}{2}} n - i + \sum_{i=\frac{n}{2}+1} ni - 1\right) \leq \frac{3n}{4}$. L'ensemble dans lequel on continue la recherche n'est pas toujours le plus gros, mais puisque le plus gros est réduit d'un facteur au moins $\frac{3}{4}$ c'est a fortiori vrai de celui qui nous interresse. En sommant sur toutes les différentes étapes, on obtiens une complexité totale inférieure à $\sum_{k=1}^{\infty} \left(\frac{3}{4}\right)^k n \leq 3n$.

4.5.2 L'arbre de Delaunay

L'arbre de Delaunay [8] est un exemple typique de construction incrémentale randomisée. L'idée de base consiste à introduire les points un à un, à mettre à jour la triangulation de Delaunay, et à se souvenir de l'historique de toutes ces modifications. Cet historique est alors utilisé comme une structure de localisation pour déterminer ou le nouveau point doit être ajouté dans la triangulation.

Plus précisément, l'arbre de Delaunay est construit de la manière suivante. A chaque triangle de Delaunay est associé une feuille de l'arbre de Delaunay, lors de l'insertion d'un nouveau point p certains triangles sont détruits et d'autres sont créés (voir figure 3.6). Si on crée un triangle pqr alors un triangle sqr dont le cercle circonscrit contenait p a été détruit par l'insertion de p. Le nœud correspondant dans l'arbre de Delaunay n'est pas détruit, il devient un nœud

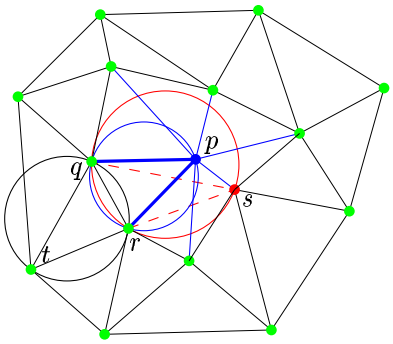

FIG. 4.12 – Dans l'arbre de Delaunay, *pqr* est le fils de *sqr* et le beau-fils de *tqr*.

interne et est déclaré *père* du nouveau nœud associé à *pqr*. La structure de l'arbre n'est pas suffisante pour assurer les localisations de nouveaux points, on ajoute donc certains liens : le voisin *tqr* de *sqr* dans la triangulation n'est pas détruit par l'insertion de p et devient voisin du nouveau triangle *pqr*, on relie ces triangles par on pointeur et on dit que *tqr* est le *beau-père* de *pqr* (voir figure 4.12).

Cette structure permet de trouver facilement de trouver tous les triangles dont le cercle circonscrit contiens un nouveau point x. En effet, imaginons que le cercle *pqr* contienne un point x, comme ce cercle est inclus dans l'union des deux cercles *sqr* et *tqr*, on en déduit que si on conneit les cercles contenant x dans la triangulation avant l'insertion de p on trouve les cercles contenant x après l'insertion de p en examinant les fils et beau-fils de ces nœuds dans l'arbre. Un simple parcours de l'arbre de Delaunay permet donc de trouver tous les nœuds de l'arbre, c'est à dire les triangles existant ou ayant existé dans la triangulation de Delaunay, dont le cercle circonscrit contient un point x.

L'analyse de cet algorithme est faite assez facilement à l'aide de l'analyse randomisé inverse (*backward randomized analysis*) [50]. Nous allons évaluer le coût de la localisation du dernier point x d'un ensemble $\mathcal{S}$ de n points insérés dans un ordre aléatoire, pour cela, il suffit de compter le nombre de nœud visité dans l'arbre de Delaunay pour localiser x, en fait on va se contenter de compter les nœuds dont le cercle contient x, pour ceux dont le cercle ne contient pas x nous renvoyons le lecteur à l'article original [8]. On considère $\mathcal{S}_k$ l'ensemble des k premiers points insérés et p le $k+1$-ème point, x sera dans le cercle d'un triangle *pqr* crée par l'insertion de p si xp est une arête de $\mathcal{DT}(\mathcal{S}_{k+1} \cup \{x\})$[1]. Puisque les points sont insérés dans un ordre aléatoire, x est un point aléatoire de $\mathcal{S}_{k+1} \cup \{x\}$ sont degré moyen est donc 6 (en utilisant la relation d'Euler) p est également un point aléatoire il a donc une probabilité $\frac{6}{k+1}$ d'être voisin de x dans cette triangulation. En sommant on obtient un coût moyen de $\sum k = 1^n \frac{6}{k+1} = O(\log n)$ pour la localisation de x. La localisation effectuée, il

[1] attention cette triangulation n'est pas calculée, elle ne sert que pour l'analyse.

reste à créer les nouveaux triangles et les nœuds correspondants, l'analyse en est encore plus simple, x étant un point aléatoire de $\mathcal{S}$, son degré dans $\mathcal{DT}(\mathcal{S})$ est en moyenne 6, il y a donc 6 nouveaux triangles crées par l'insertion du n-ième point et ils sont créés en temps $O(1)$. En sommant maintenant sur n, on a un coût moyen de $O(N \log N)$ pour la construction complête de la triangulation.

4.5.3 Autres algorithmes randomisés pour Delaunay

De nombreux autres algorithmes pour la triangulation de Delaunay utilisent plus ou moins la randomisation.

Guibas, Knuth et Sharir [31] on proposé un algorithme assez similaire. L'insertion d'un point est effectuée en coupant le triangle qui le contient en trois, puis en effectuant quelques bascules de diagonales (paragraphe 3.4.3). On conserve tous les triangles ainsi construits pour effectuer la localisation. L'analyse est très similaire à celle l'arbre de Delaunay et on aboutit également à une complexité moyenne de $O(n \log n)$ si les points sont insérés dans un ordre aléatoire. Il est également possible d'utiliser le même genre de graphe pour des diagrammes de Voronoï plus généraux [37].

Ces structures conservant l'histoire de la construction de la triangulation peuvent être généralisées pour permettre la supression de points. On a alors deux alternatives. Tout d'abord on peut se rappeller de l'histoire complète de la construction [48], le coût d'une localisation dépend alors de la taille de l'histoire et non plus du nombre de points effectivement présent dans la triangulation. Une autre solution consiste à reconstruire une histoire «comme si le point supprimé n'avait jamais été inséré» [20].

Mulmuley a proposé un algorithme ou la randomisation ne repose pas uniquement sur l'ordre d'insertion des points [43] ce qui permet de minimiser la probabilité d'être dans un mauvais cas, la complexité moyenne est optimale en $O(n \log n)$. L'algorithme incrémental avec marche (paragraphe 3.4.2) peut également être amélioré en choisissant un bon point de départ pour la marche [42, 22] grace à un échatillon aléatoire, on a alors une complexité de $O(n \sqrt[3]{n})$ pour des points uniformément répartis. La hiérarchie de Delaunay [18] construit une succession d'échatillons aléatoires, la localisation se fait en marchant à partir d'un sommet d'un échantillon, on obtiens alors une complexité optimale en $O(n \log n)$ sans hypothèse sur la répartirion des points. Ces deux derniers algorithmes sont repris au chapitre 8.3.

Enfin, comme on l'a annoncé au paragraphe 4.1, la triangulation d'un polygone convexe peut être réalisée très simplement si on se contente d'une complexité randomisée [13] alors que le seul algorithme déterministe connu est assez compliqué [1]. Dans les algorithmes randomisés cidessus, c'est la localisation du nouveau point à insérer qui produit une complexité logarithmique, le coût de la modification de la triangulation étant lui constant en moyenne comme l'a montré l'analyse inverse. On va donc montrer que la connsaissance du polygone convexe permet d'économiser la partie localisation [13]. Étant donné un polygone convexe $P = p_1 p_2 \dots p_n$, on choisi un sommet aléatoire p_i, on calcule récursivement $\mathcal{DT}(P \setminus \{p_i\})$. Maintenant on sait bien que le point p_i sera relié à

p_{i-1} et à p_{i+1} dans la triangulation finale, il n'est donc pas nécessaire de chercher où est p_i dans $\mathcal{DT}(P \setminus \{p_i\})$ mais d'inserer directement p_i en partant du triangle ayant $p_{i-1}p_{i+1}$ comme arête.

Chapitre 5

Prédicats géométriques

5.1 Définition de la notion de prédicat

Comme nous l'avons vu, la description d'un algorithme géométrique est essentiellement combinatoire, mais les décisions prises à l'intérieur d'un tel algorithme reposent sur des tests géométriques «élémentaires» que nous appellerons prédicats.

Un prédicat géométrique est une question géométrique élémentaire admettant un nombre fini de réponses. En général, un prédicat se réduit à l'évaluation du signe d'une certaine expression et admet donc trois réponses ; dans certains cas un peu plus complexes, on peut être amené à évaluer le signe de plusieurs expressions avant de pouvoir trouver le résultat.

Dans le cas qui nous préoccupe ici de la triangulation de Delaunay, les prédicats utilisés sont, bien sur, les prédicats de cocyclicité de 4 points et d'orientation de 3 points. Il faut cependant noter que beaucoup d'algorithmes liés à la triangulation de Delaunay peuvent utiliser des prédicats plus complexes.

5.2 Conception d'un prédicat

Le livre entier pourrait être consacré à cette notion de prédicat et à l'énumération des prédicats utiles en géométrie, nous nous bornerons ici à donner les grandes lignes de la conception d'un prédicat en l'illustrant sur l'exemple du prédicat de cocircularité.

5.2.1 Aspect géométrique

Un prédicat a tout d'abord une signification géométrique, dans le cas de notre exemple, on cherche à savoir si le point t est à l'intérieur, à l'extérieur ou sur le bord du cercle circonscrit au triangle positivement orienté pqr (voir figure 5.1). La définition géométrique du prédicat doit bien préciser quelles sont

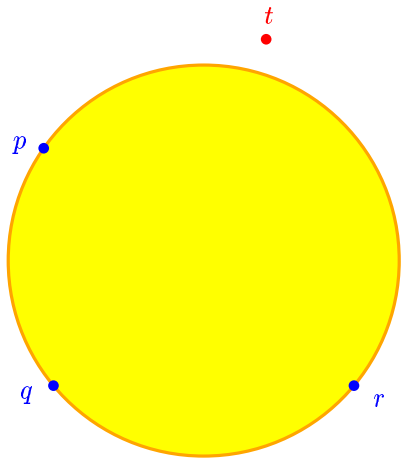

FIG. 5.1 – Le prédicat de cocyclicité.

les données du problèmes, par exemple si on voulait tester t par rapport à un cercle défini par son centre et son rayon, on aurait un prédicat différent.

5.2.2 Aspect algébrique

Une fois cette question posée, on cherche en général à obtenir une formulation algébrique de notre problème. Assez souvent, notre prédicat se réduit à évaluer le signe d'une certaine expression dépendant des coordonnées des points ou des coefficients définissant les objets géométriques intervenant dans notre problème.

Dans notre exemple, le test de cocyclicité consiste à évaluer le signe du déterminant

$$\begin{vmatrix} x_p & x_q & x_r & x_t \\ y_p & y_q & y_r & y_t \\ x_p^2+y_p^2 & x_q^2+y_q^2 & x_r^2+y_r^2 & x_t^2+y_t^2 \\ 1 & 1 & 1 & 1 \end{vmatrix}.$$

La formulation algébrique n'est en général pas unique et trouver la plus simple n'est pas toujours chose aisée. On peut la chercher «à la main» ou utiliser des outils de géométrie algébrique [19].

Si plusieurs formulations algébriques sont possibles, plusieurs critères de complexité peuvent être pris en compte en fonction des choix arithmétiques détaillés au paragraphe suivant.

5.2.3 Aspect arithmétique

(`paragraphe à finaliser :=)` Une fois la formule établie, il reste à l'évaluer. Pour cela il faut préciser quel est le type des nombres que l'on manipule. En effet les ordinateurs permettent de manipuler différentes sortes de nombres

tels que nombres entiers, nombres rationnels, nombre flottants. Nous détaillons tout cela dans le prochain chapitre.

Degré

Taille de la formule

5.2.4 Autres prédicats utiles pour Delaunay

(`paragraphe à finaliser :=`)

Orientation d'un triangle,

Comparaison de puissances.

Fortune.

Chapitre 6

Les problèmes de précision numérique

6.1 L'arithmétique de la machine

Le concept de nombre réel est une pure abstraction mathématique et ne saurait être représenté dans un ordinateur qui ne dispose que d'une mémoire finie. On peut alors opter pour différentes solutions. Toutes ces solutins reviennent à travailler avec un sous-ensemble fini de nombres,les différences résident dans les classes utilisées pour choisir ces nombres et la manière de les décrire.

6.1.1 Nombres entiers

Une première classe de nombres est celle des nombres entiers. On travaille ici avec un ensemble mathématique bien défini et aux propriétées bien connues. La représentation informatique impose cependant quelques limitations.

On utilise classiquement une représentation des nombres entiers par 32 (ou eventuellement 64) chiffres en base 2, permettant ainsi de manipuler des entiers de l'intervalle $[-2^{32}, 2^{32}]$ (il faut donc gérer les débordements à l'extérieur de cet intervalle).

On peut également utiliser les nombres *en virgule flottante* de la machine pour représenter des entiers. En effet un type de nombre `double` avec une mantisse sur 53 bits permettra une représentation exacte des entiers de l'intervalle $[-2^{53}, 2^{53}]$ avec un gestion plus aisée des débordements hors de cet intervalle.

On peut enfin utiliser des librairies permettant de gérer des entiers de longueur arbitraire. Le temps de calcul se ressent evidemment.

6.1.2 Nombres en virgule flottante

C'est l'approche informatique la plus classique. Plutot que d'utiliser des nombres réels on va se contenter d'une approximation. Cette approche est

conforme au calcul numérique que l'on peut faire à la main dans lequel un nombre est représenté sous la forme d'un nombre décimal. Ici on utilise une représentation avec des chiffres binaires. La norme IEEE 754 [35] spécifie exactement les formats possibles pour représenter de tels nombres ainsi que les règles d'arrondis à appliquer pour obtenir le résultat d'un calcul.

Représentation

Un nombre flottant double précision (`double` en C) est représenté par 64 bits. Sur ces 64 bits se répartissent comme suit : 1 bit s, 52 bits m et 11 bits e représentent le nombre $(-1)^s \cdot 0.1m \cdot 2^{e-1024}$ en notation normalisée. La notation normalisée signifie que l'on choisit l'exposant pour que la mantisse soit comprise entre $\frac{1}{2}$ et 1, son développement binaire commence donc toujours par 0.1 et il n'est pas nécessaire de représenter ces deux chiffres.

Les nombres représentables en machines sont donc régulièrement espacé avec des intervalles de 2^{p-52} entre 2^p et 2^{p+1}.

Lorsque l'on se rapproche de zéro cette notation permet donç lorsque $e = 0$ de représenter les nombres de $[2^{-1025}, 2^{-1024}]$ avec des écarts de 2^{-1077} mais pas du tout les nombres entre 0 et 2^{-1025} pour pallier à ce trou dans les nombres représentables on utilise lorsque $e = 0$ la représentation dénormalisée. Si $e = 0$, la valeur du nombre représentée est $(-1)^s \cdot 0.m \cdot 2^{-1024}$ on peut ainsi représenter les nombres de l'intevalles $[0, 2^{-1024}]$ avec des écarts réguliers de 2^{-1076}.

Règles d'arrondis

Si on a deux nombres flottants `a` et `b` le résultat d'une opération arithmétique sur ces nombres, la multiplication pour fixer les idées, n'est en général pas un nombre représentable. Si on appelle s et s' les deux nombres représentables qui encadrent la vrai valeur de `a·b`.

La valeur calculée par la machine comme le résultat de `a*b` est alors s ou s' ($s < s'$) selon le mode d'arrondi choisi. La norme IEEE 754 propose quatre modes d'arrondi : au plus près, vers $+\infty$, vers $-\infty$ et vers 0.

En mode «au plus près» `a*b` est arrondi au plus proche flottant représentable, il faut donc comparer la valeur exacte `a·b` avec $\frac{s+s'}{2}$. En mode vers $+\infty$ on utilise s'. En mode vers $-\infty$ on utilise s. Et en mode vers 0 on utilise s si `a·b`< 0 et s' sinon (voir figure 6.1).

Il important de noter que la notion d'arrondi est définie très précisemment. La valeur utilisée pour `a*b` est parfaitement défini par la norme afin d'assurer la portabilité sur différents processeurs.

Cette norme définissant l'arrondi avec précision s'applique au quatre opérations `+,-,*,/` et à la racine carré. Il faut parfois utiliser certaines options de compilation pour que le processeur se conforme à la norme.

Des travaux pour étendre ce type de norme définissant l'arrondi avec précision à d'autres fonctions telles que sinus, exponentielle... existent. La difficulté résidant dans le fait qu'il ne suffit pas de calculer une approximation précise de la valeur recherchée, mais qu'il faut comparer *exactement* cette valeur avec les

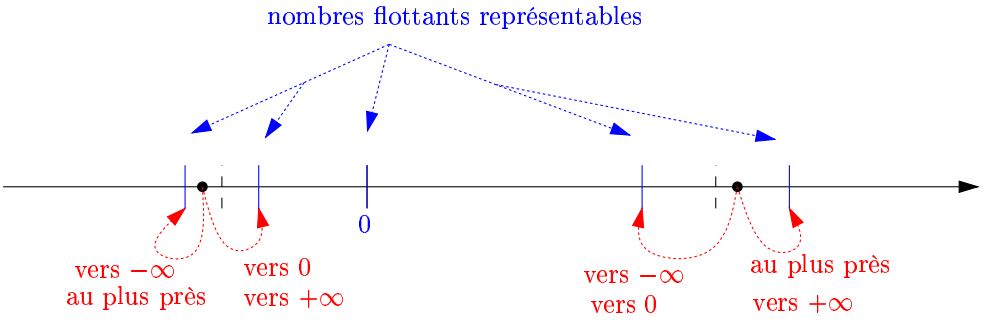

FIG. 6.1 – Les quatre modes d'arrondi.

bornes des intervalles ce qui peut s'avérer très délicat quand on est très près de ces bornes.

Propriétés

Les propriétés de l'arithmétique machine ne sont pas celle des réels, la propriété la plus cruciale étant l'associativité.

Commutativité L'addition et la multiplication sont commutatives.
`a+b=b+a` et `a*b=b*a`.

Monotonicité L'arrondi préserve l'ordre respectif de deux nombres, toutefois les inégalitées strictes deviennent large. En effet il est clair que deux nombres différents peuvent avoir le même arrondi.
`a·b` $<$ `c·d` $\Longrightarrow$ `a*b` $\leq$ `c*d`.

Compatibilité soustraction et comparaison La soustraction et la comparaison sont bien compatible on a `a==b` $\Longleftrightarrow$ `a-b == 0`

Cette propriété est lié à l'usage des nombres dénormalisé qui guaranti que l'écart entre deux nombres représentables est également un nombre représentable.

Exactitude Comme on l'a vu, le résultat d'un calcul est toujours arrondi à un nombre représentable, mais si on peut garantir que le résultat exact est représentable, alors les nombres flottants font en fait du calcul exact. Les flottants double précision font ainsi d'excellents entiers sur 53 bits avec les avantages suivants :
— gestion plus facile des débordements (hors de l'intervalle $[-2^{53}, 2^{53}]$). En effet le calcul sur les entiers machines est fait modulo le plus grand entier, il n'est donc pas immédiat de savoir si une multiplication a débordé.
— possibilité de travailler dans les multiples de 2^k ($k \in \mathbb{Z}$) directement, sans

faire explicitement les multiplications par 2^{-k} pour repasser aux entiers.
— sur certaines architectures, le calcul sur les flottants double précision est plus rapide que sur les entiers 64 bits voire même plus rapide que les entiers 32 bits.

Associativité Le caractère associatif est perdu, les bits les moins significatifs du résultat d'une opération disparraissent.

On peut parfois utiliser cette non associativité, par exemple l'arrondi entier d'un nombre `a`$\in [-252, 2^{52}]$ peut être obtenu par `Ea = (a + CST) - CST` où `CST` vaut $3 \cdot 2^5 2$. L'addition a pour effet la disparition des bits de poids faible de `a`, la soustraction est exacte. On peut ensuite récupérer l'erreur par `da = a - Ea`, cette soustraction sera calculée exactement.

Arithmétique 2 décimales.

Afin d'illustrer certains problème lié à l'arithmétique flottante de manière plus intuitive, nous utiliserons un modèle d'arithmétique à deux chifres décimaux significatifs.

Dans une telle arithmétique on a :

`38 + 45`$= 83 \rightarrow$`83`	Résultat exact sur deux chiffres
`38 + 86`$= 124 \rightarrow$`130`	Résultat arrondi vers $+\infty$
`38 + 86`$= 124 \rightarrow$`120`	Résultat arrondi au plus près, vers 0 ou vers $-\infty$
`38-((38+86)-86)=38-(120-86)=38-34=4`	erreur de l'addition ci dessus.

6.1.3 Calcul symbolique

Si les nombres entiers sont insuffisants et que l'on ne désire pas utiliser des approximations comme les nombres flottants, il reste la solution du calcul symbolique.

Les nombres ont alors une représentation symbolique de leur valeur exacte. Selon les opérations supportées on peut utiliser des nombres rationnels ou des nombres algébriques.

6.2 Erreurs liées au calcul flottant

(`paragraphe à finaliser :=`) Illustration d'incohérences géométriques liées au calcul flottant. Exemple avec l'arithmétique à deux chiffres en base 10 : quatre points $pqrs$ tel que 1– pqr direct en réalité mais indirect par le test en calcul arrondi (la machine peut calculer "faux") 2– pqs, qrs, rps direct par la calcul comme en réalité ce qui est incohérent avec le fait que pqr est calculé indirect.

6.3 Première solution : la vérification de cohérence

(`paragraphe à finaliser :=`) Algorithme style Sugihara.

6.4 Deuxième solution : le calcul exact

Solution apparemment résolvant tout les problèmes. Mais

- Ça coûte cher 1) par opération. 2) le coût d'une opération dépends de la taille des nombres manipulés(change l'étude de complexité). 3) solutions filtre, différentes méthodes de calcul exact.
- Les cas dégénérés deviennent inévitables.
- Ça ne marche que si l'on sait calculer exactement les prédicats intéressants

(`paragraphe à finaliser :=)` Pour Delaunay, profondeur bornées. Différentes solutions exactes, grands entiers, Shewchuk, LN, "ad'hoc". Efficacité des filtres.

Chapitre 7

Les problèmes de dégénérescences

(paragraphe à finaliser :=)

7.1 Description des algorithmes

L'hypothèse sans dégénérescence supprime tous les cas particuliers. Si il y a des dégénérescences, il faut dire ce que l'on fait dans ces cas particuliers.

Pour Delaunay, que ce passe-t-il si trois points sont alignés. Si quatre points sont cocycliques. Diagramme de Delaunay, non unicité de la triangulation de Delaunay.

7.2 Analyse des algorithmes

Il peut y avoir des influences sur l'analyse des algorithmes. Exemple : l'arbre de Delaunay. [Otfried,Herve]

7.3 Première solution : le traitement des cas particuliers

Décrire ce que l'on fait dans chaque cas particulier, analyse adaptée.

7.4 Deuxième solution : les méthodes de perturbation

[EM,CE,ADS] Tous ce passe comme si il n'y avait pas de cas dégénérés. En particulier, unicité de la triangulation. Pas de problèmes d'analyse.

Chapitre 8

Algorithmes

(paragraphe à finaliser :=)

8.1 "Buckets"

[Dwyer] [Su Drysdale]

8.2 Division fusion

Étude probabiliste et pratique montrant que ça marche bien (si les points sont connus a l'avance). Moreau Lemaire / Shewchuk

8.3 Algorithmes incrémentaux

Algorithme de [MSZ] en $O(n^{\frac{4}{3}})$, principe et complexité pour des points aléatoires. Amélioration en $O(n^{\frac{5}{4}})$. Moreau Lemaire

Hiérarchie de triangulations [D]

Chapitre 9

Recalage

(`paragraphe à finaliser :=`)
Selon l'avancement des choses ? Au moins mentionner le problème quelque part.

Chapitre 10

Requêtes

(`paragraphe à finaliser :=`)
Delaunay est une bonne structure pour faire des requêtes en pratique.

Chapitre 11

Delaunay et les segments

(paragraphe à finaliser :=)

11.1 Voronoï de segments

(paragraphe à finaliser :=)

11.2 Delaunay contraint

(paragraphe à finaliser :=)

11.3 Triangulation conforme

(paragraphe à finaliser :=)

Chapitre 12

Dans l'espace

(paragraphe à finaliser :=)

Chapitre 13

Une application : reconstruction tridimensionnelle

Diagramme de Voronoï et triangulation de Delaunay ont de nombreuses applications (voir parapgarphe 3.2), nous allons ici donner quelques détails sur quelques applications à la reconstruction d'objets en deux ou trois dimensions.

Le problème de base est étant donné un ensemble de points $\mathcal{S}$ pris sur le bord d'un ou plusieurs objets, reconstruire une triangulation dont la surface de ces objets. En absence d'autres hypothèses, il est logique d'utiliser des relations de proximités entre les points pour savoir qui relier à qui. D'où l'utilité de la triangulation de Delaunay qui justement intègre, par le biais des critères de sphères vides, de telles notion de proximité. Pour toutes ces méthodes, il faut bien sur que l'échantillonage de la surface recherchée par les points de $\mathcal{S}$ soit suffisant, on ne peut pas reconstruire des choses qui ne serait absolument pas présente dans $\mathcal{S}$. Comme dans tous les algorithmes de géométrie, on recherche ici essentiellement la topologie, la combinatoire de l'objet à reconstruire, dans une étape ultérieure on peut améliorer la reconstruction en lissant la forme obtenue pour la rendre moins polyhédrique (rendu de Gouraud, *splines*, *snakes*...).

13.1 Graphes à définition locale

Pour approximer la forme de l'objet défini par les points de $\mathcal{S}$ on va utiliser un ensemble d'arêtes en deux dimensions ou de triangles en trois dimensions reliant les points de $\mathcal{S}$. Un certain nombre de graphes utilisé ont une définition purement locale.

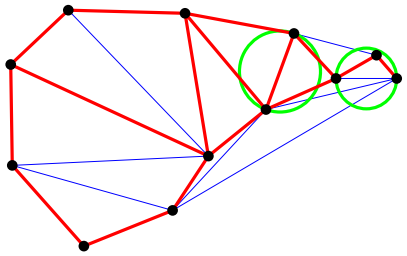

FIG. 13.1 – Exemple de graphe de Gabriel

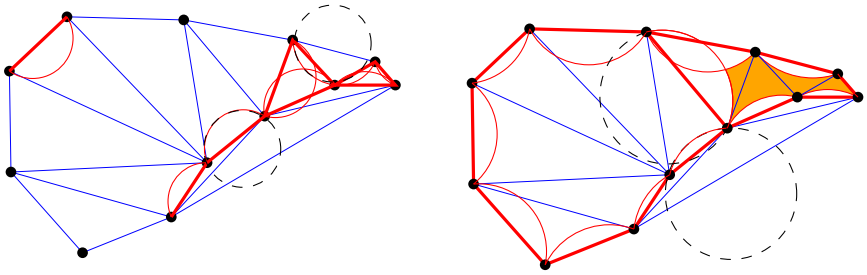

FIG. 13.2 – Exemple d'α formes et d'α squelettes.

13.1.1 Graphe de Gabriel

Deux points de $\mathcal{S}$, p et q définissent une arête du graohe de Gabriel $\mathcal{G}(\mathcal{S})$ si et seulement si le cercle (la sphère) de diamètre pq ne contient pas de point de $\mathcal{S}$. $\mathcal{G}(\mathcal{S})$ est clairement inclus dans $\mathcal{DT}(\mathcal{S})$ et peut être extrait de $\mathcal{DT}(\mathcal{S})$ en temps linéaire. Un exemple est donné par la figure 13.1.

Compte tenu du théorème de l'arc capable, il est facile de voir que le graphe de Gabriel va conserver une arête pq de Delaunay si les deux triangles de Delaunay adjacents pqr et pqr' sont aigus en r et r'.

13.1.2 Alpha formes

Une première approximation très grossière de la forme recherchée est l'enveloppe convexe de $\mathcal{S}$. On peut construire l'enveloppe convexe de la façon suivante : on part avec le plan tout entier, puis on dispose d'un outil permettant d'enlever des demi-plans. Si on enlève des demi-plans tant que cela est possible sans toucher au points de $\mathcal{S}$ on obtient à la fin l'enveloppe convexe. Si maintenant notre outil est plus sophistiqué, et si au lieu d'enlever des demi-plans il permet d'enlever des disques de rayon $1/\alpha$ on obtient l'alpha forme.

Si le cercle passant par p et q de rayon $1/\alpha$ ne contient aucun point de $\mathcal{S}$, le segment pq appartient à l'alpha squelette.

L'alpha squelette modelisera bien la courbe recherchée

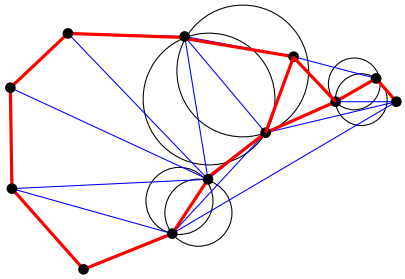

FIG. 13.3 – Exemple de β squelette.

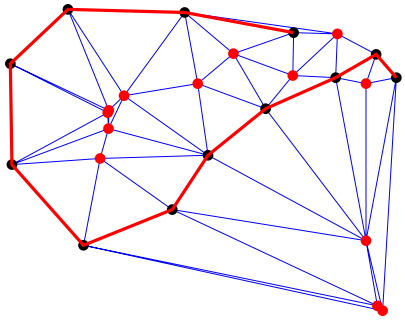

FIG. 13.4 – Exemple de *crust*.

- si le rayon du cercle est suffisamment faible pour rentrer dans les concavités (i.e. α est supérieure à la courbure de la courbe),
- si l'echantillonage est suffisant pour que le cercle ne puisse pas passer entre deux points consécutifs sur la courbe,
- si la courbe ne comporte pas d'etranglements.

13.1.3 Beta squelettes

Une arête pq fait partie du β squelette ($\beta \geq 1$) si l'union des deux disques de rayon $\beta|pq|/2$ passant par p et q est vide. Si $\beta = 1$ on obtient exactement le graphe de Gabriel. Si β augmente les arêtes du β squelette sont toujours des arêtes de Delaunay, mais on en rejette de plus en plus.

13.2

13.2.1 Crust

(`paragraphe à finaliser :=`)

Chapitre 14

Autres problèmes de géométrie

(`paragraphe à finaliser :=`)

Dire que les grands principes exposés pour Delaunay trouvent également leur application dans d'autres problèmes de géométrie.

Généralisation de Voronoï enveloppe convexe, arrangement de courbes ...

Bibliographie

[1] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6) :591–604, 1989.

[2] F. Aurenhammer. Power diagrams : properties, algorithms and applications. *SIAM J. Comput.*, 16 :78–96, 1987.

[3] F. Aurenhammer and H. Imai. Geometric relations among Voronoi diagrams. *Geom. Dedicata*, 27 :65–75, 1988.

[4] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9) :643–647, September 1979.

[5] M. Bern, S. Mitchell, and J. Ruppert. Linear-size nonobtuse triangulation of polygons. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 221–230, 1994.

[6] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in CGAL. *Comput. Geom. Theory Appl.*, 22 :5–19, 2002.

[7] Jean-Daniel Boissonnat, Olivier Devillers, René Schott, Monique Teillaud, and Mariette Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8 :51–71, 1992.

[8] Jean-Daniel Boissonnat and Monique Teillaud. On the randomized construction of the Delaunay tree. *Theoret. Comput. Sci.*, 112 :339–354, 1993.

[9] Jean-Daniel Boissonnat and Mariette Yvinec. *Géométrie Algorithmique.* Ediscience international, Paris, 1995.

[10] Prosenjit Bose and Luc Devroye. Intersections with random geometric objects. *Comput. Geom. Theory Appl.*, 10 :139–154, 1998.

[11] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5) :485–524, 1991.

[12] Bernard Chazelle, Olivier Devillers, Ferran Hurtado, Mercè Mora, Vera Sacristán, and Monique Teillaud. Splitting a Delaunay triangulation in linear time. *Algorithmica*, 34 :39–46, 2002.

[13] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1986.

[14] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4 :387–421, 1989.

[15] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry : Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[16] Olivier Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Internat. J. Comput. Geom. Appl.*, 2(1) :97–111, 1992.

[17] Olivier Devillers. An introduction to randomization in computational geometry. *Theoret. Comput. Sci.*, 157 :35–52, 1996.

[18] Olivier Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[19] Olivier Devillers, Alexandra Fronville, Bernard Mourrain, and Monique Teillaud. Algebraic methods and arithmetic filtering for exact predicates on circle arcs. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 139–147, 2000.

[20] Olivier Devillers, Stefan Meiser, and Monique Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. *Comput. Geom. Theory Appl.*, 2(2) :55–80, 1992.

[21] Olivier Devillers, Stefan Meiser, and Monique Teillaud. The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. Report 1620, INRIA Sophia-Antipolis, Valbonne, France, 1992.

[22] Luc Devroye, Ernst Peter Mücke, and Binhai Zhu. A note on point location in Delaunay triangulations of random points. *Algorithmica*, 22 :477–482, 1998.

[23] M. T. Dickerson and R. S. Drysdale. Fixed-radius near neighbor search algorithms for points and segments. *Inform. Process. Lett.*, 35 :269–273, 1990.

[24] H. Djidjev and A. Lingas. On computing Voronoi diagrams for sorted point sets. *Internat. J. Comput. Geom. Appl.*, 5 :327–337, 1995.

[25] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[26] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete Comput. Geom.*, 1 :25–44, 1986.

[27] Herbert Edelsbrunner, Tiow Seng Tan, and Roman Waupotitsch. $O(N^2 \log N)$ time algorithm for the minmax angle triangulation. *SIAM J. Sci. Statist. Comput.*, 13(4) :994–1008, July 1992.

[28] J. Erickson and R. Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discrete Comput. Geom.*, 13 :41–57, 1995.

[29] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2 :153–174, 1987.

[30] Paul-Louis George and Houman Borouchaki. *Triangulation de Delaunay et maillage. Applications aux éléments finis.* Hermes, Paris, France, 1997.

[31] Leonidas J. Guibas, D. E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7 :381–413, 1992.

[32] Leonidas J. Guibas and J. Stolfi. Ruler, compass and computer : the design and analysis of geometric algorithms. In R. A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, volume 40 of *NATO ASI Series F*, pages 111–165. Springer-Verlag, 1988.

[33] F. Hurtado and M. Noy. Graph of triangulations of a convex polygon and tree of triangulations. *Comput. Geom. Theory Appl.*, 13 :179–188, 1999.

[34] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete Comput. Geom.*, 22(3) :333–346, 1999.

[35] *IEEE Standard for binary floating point arithmetic, ANSI/IEEE Std* 754 – 1985. New York, NY, 1985. Reprinted in SIGPLAN Notices, 22(2) :9–25, 1987.

[36] K. Kedem and Micha Sharir. An efficient motion planning algorithm for a convex rigid polygonal object in 2-dimensional polygonal space. *Discrete Comput. Geom.*, 5 :43–75, 1990.

[37] Rolf Klein, Kurt Mehlhorn, and Stefan Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput. Geom. Theory Appl.*, 3(3) :157–184, 1993.

[38] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Math. Software III*, pages 161–194. Academic Press, New York, NY, 1977.

[39] Christophe Lemaire and Jean-Michel Moreau. Analysis of a class of k–dimensional merge procedures, with an application to $2d$ Delaunay triangulation in expected linear time after two–directional sorting. In *Proc. 9th Canad. Conf. Comput. Geom.*, pages 129–134, 1997.

[40] Kurt Mehlhorn. *Data Structures and Algorithms 3 : Multi-dimensional Searching and Computational Geometry*, volume 3 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Heidelberg, Germany, 1984.

[41] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, New York, NY, 1995.

[42] Ernst P. Mücke, Isaac Saias, and Binhai Zhu. Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 274–283, 1996.

[43] K. Mulmuley. Randomized multidimensional search trees : Dynamic sampling. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 121–131, 1991.

[44] K. Mulmuley. *Computational Geometry : An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[45] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tessellations : Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, UK, 1992.

[46] F. P. Preparata and M. I. Shamos. *Computational Geometry : An Introduction.* Springer-Verlag, New York, NY, 1985.

[47] A. Schönhage, M. Paterson, and N. Pipinger. Finding the median. *J. Comput. Syst. Sci.*, 13 :184–199, 1976.

[48] O. Schwarzkopf. Dynamic maintenance of geometric structures made easy. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 197–206, 1991.

[49] R. Seidel. A method for proving lower bounds for certain geometric problems. In G. T. Toussaint, editor, *Computational Geometry*, pages 319–334. North-Holland, Amsterdam, Netherlands, 1985.

[50] R. Seidel. Backwards analysis of randomized geometric algorithms. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 37–68. Springer-Verlag, 1993.

[51] J. R. Shewchuk. Triangle : Engineering a 2d quality mesh generator and Delaunay triangulator. In *First Workshop on Applied Computational Geometry.* Association for Computing Machinery, May 1996.

[52] V. Turau. Fixed-radius near neighbors search. *Inform. Process. Lett.*, 39 :201–203, 1991.

Résumé

Nous proposons à travers l'exemple de la triangulation de Delaunay une introduction à la géométrie algorithmique et au calcul géométrique.

La géométrie algorithmique «classique» propose des solutions (`paragraphe à finaliser :=)`