INF562
Introduction à la géométrie algorithmique et ses applications

Luca Castelli Aleardi et Steve Oudot

février 2020

Table des matières

I Les fondamentaux de la géométrie algorithmique **1**

1 Géométrie discrète et algorithmique: introduction **3**
1.1 Graphes, cartes et maillages . . . 3
1.1.1 Graphes . . . 3
1.1.2 Cartes et subdivisions planaires . . . 3
1.1.3 Arbres . . . 5
1.1.4 Triangulations . . . 5
1.1.5 Complexes et maillages . . . 5
1.2 Formule d'Euler en dimension 2 . . . 6
1.3 Représentations de données géométriques . . . 9
1.3.1 Structures de données abstraites pour la géométrie . . . 9
1.3.2 Quelques structures de données pour les maillages . . . 9
1.3.2.1 Maillages surfaciques (polygonaux): Half-edge . . . 10
1.3.2.2 Une structure de données pour les triangulations . . . 10
1.3.2.3 Quad-edge data structure . . . 10
1.4 Un problème classique: localisation planaire . . . 11
1.5 Géométrie convexe . . . 13
1.5.1 Coordonnées barycentriques . . . 13
1.5.2 Enveloppes affines, enveloppes convexes . . . 13
1.5.3 Théorèmes fondamentaux: Radon, Helly et Carathéodory . . . 14
1.5.4 Existence de center points . . . 16
1.5.5 Combinatoire des Polytopes . . . 17
1.6 Enveloppes convexes en 2D: algorithmes et complexité . . . 17
1.6.1 Algorithme de Jarvis . . . 18
1.6.2 Approche par balayage : l'algorithme de Graham-Andrew . . . 18
1.6.3 Approche par *Division-Fusion* . . . 20
1.6.4 Borne inférieure sur la complexité . . . 22
1.7 Enveloppes convexes en dimension supérieure . . . 22
1.8 Arrangements de droites dans le plan . . . 23
1.9 Robustesse des algorithmes géométriques . . . 24

2 Triangulations de Delaunay et diagrammes de Voronoï **29**
2.1 Définitions . . . 29
2.1.1 Triangulations de Delaunay . . . 30
2.1.2 Diagrammes de Voronoï . . . 31
2.2 Quelques propriétés remarquables . . . 31
2.2.1 Localement Delaunay contre globalement Delaunay . . . 31
2.2.2 Optimisation de l'angle minimal et de la séquence des angles . . . 33
2.2.3 Arbre couvrant minimal . . . 34

2.3 Taille et calcul de la triangulation de Delaunay . . . 35
2.3.1 Taille de la triangulation . . . 35
2.3.2 Calcul de la triangulation . . . 36
2.4 Extensions et notes bibliographiques . . . 39

3 Aspects géométriques des graphes (planaires) 43
3.1 Introduction et motivations . . . 43
3.2 Graphes: le point de vue combinatoire . . . 43
3.2.1 Cartes (planaires ou plongées sur des surfaces) . . . 43
3.2.2 Graphes et 3-connexité. . . . 45
3.3 Deux célèbres caractérisations des graphes planaires . . . 46
3.3.1 Théorème de Kuratowski . . . 46
3.3.2 Théorème de Koebe . . . 47
3.4 Théorie algébrique des graphes . . . 51
3.5 Dessin de graphes planaires . . . 52
3.5.1 Représentation d'un graphe dans $\mathbb{R}^d$ . . . 52
3.5.2 Représentations convexes et méthode de Tutte . . . 53
3.5.2.1 Matrice laplacienne et méthode de Tutte . . . 55
3.5.3 Dessiner un graphe en minimisant son énérgie . . . 57
3.5.3.1 Dessin spectral . . . 57
3.5.4 Dessins sur une grille et méthode de Schnyder . . . 59
3.5.5 Preuve du Théorème barycentrique de Tutte (Lemme 3.21) . . . 61
3.5.6 Notes bibliographiques . . . 63
3.6 Graphes et petits séparateurs . . . 63
3.6.1 Séparateurs et graphes planaires . . . 64
3.6.2 Séparateurs géométriques . . . 66
3.7 Applications algorithmiques des séparateurs . . . 68
3.7.1 Localisation planaire . . . 68
3.7.2 Compression et codage de graphes . . . 69

II Les développements plus récents 73

4 Reconstruction de formes géométriques 75
4.1 Reach d'une forme géométrique . . . 76
4.2 Triangulation de Delaunay restreinte . . . 79
4.3 Complexe de témoins et reconstruction multi-échelles . . . 81
4.3.1 Le complexe de témoins . . . 82
4.3.2 L'algorithme de reconstruction multi-échelles . . . 84
4.3.3 Correction de l'algorithme . . . 85
4.4 Extensions et notes bibliographiques . . . 88

5 Problèmes de proximité 93
5.1 Localisation planaire en petite dimension . . . 93
5.1.1 Méthode du Slab . . . 93
5.1.2 Triangulations hiérarchiques . . . 94
5.2 Range searching et recherche du plus proche voisin . . . 96
5.2.1 Range trees . . . 97
5.2.1.1 Orthogonal Range Search en dimension supérieure . . . 99
5.2.1.2 Amélioration: Fractional Cascading . . . 99

5.2.2 kd-trees: plus proche voisin 100

6 Algorithmes d'approximation géométrique 105
6.1 Quelques mots sur la complexité des algorithmes 105
6.2 Le problème du voyageur de commerce 107
6.2.1 Metric-TSP 108
6.2.2 Euclidean-TSP 110
6.2.2.1 Renormalisation et projection 111
6.2.2.2 Construction du quadtree 111
6.2.2.3 Mise en place des portails 112
6.2.2.4 Calcul du plus court cycle polygonal respectant les portails . . . 112
6.2.2.5 Modification du cycle polygonal en cycle hamiltonien sur P . . . 116
6.2.2.6 Final 116

Table des figures

1.1 Graphes et cartes planaires 4
1.2 Subdivisions planaires 5
1.3 Triangulations 6
1.4 Formule d'Euler pour les graphes planaires 7
1.5 Opérations d'Euler 9
1.6 Représentation avec demi-arêtes 10
1.7 Représentation de maillages triangulaires 11
1.8 Quad-Edge data structure 12
1.9 Localisation par marche aléatoire 12
1.10 Coordonnées barycentriques 13
1.11 Théorème de Radon 14
1.12 Théorème Helly 15
1.13 Théorème du center point 16
1.14 Enveloppe convexe: algorithme par balayage de Graham-Andrew 19
1.15 Enveloppe convexe 21
1.16 Enveloppe convexe 3D 23
1.17 Exécutions erronées de l'algorithme de Jarvis 25

2.1 Diagramme de Voronoï et triangulation de Delaunay duale 32
2.2 Pour les preuves des théorèmes 2.7 et 2.9 33
2.3 Insertion d'un sommet dans une triangulation de Delaunay 37
2.4 Zone de conflit 38

3.1 Maillages, surfaces discrètes et graphes planaires 44
3.2 Cartes combinatoires 45
3.3 Graphes et cartes planaires 46
3.4 Cartes planaires enracinées 46
3.5 Représentation de Koebe-Andreev-Thurston de graphes planaires 47
3.6 Théorème de Koebe 48
3.7 Preuve du théorème de Koebe 49
3.8 Matrice d'adjacence et matrice laplacienne 51
3.9 Représentation d'un graphe 52
3.10 Méthode barycentrique de Tutte 53
3.11 Théorème barycentrique de Tutte 55
3.12 Dessin spectral de graphes 58
3.13 Forêts de Schnyder 60
3.14 Interprétation géométrique des forêts de Schnyder 60
3.15 Séparateurs de graphes planaires 64
3.16 Séparateurs de graphes planaires 65

3.17 Théorème de Koebe sur la sphère 66
3.18 Séparateurs géométriques et théorème de Koebe 67

4.1 Processus de numérisation d'un objet en 3d 75
4.2 Quelques formes géométriques et leurs axes médians 77
4.3 Quelques formes géométriques à reach positif ou nul 78
4.4 Delaunay restreint à une courbe plane 80
4.5 Quelques nuages de points représentant des structures multi-échelles 81
4.6 Séquence de reconstructions produite par l'algorithme de Guibas et Oudot 82
4.7 Définition du complexe de témoins 83
4.8 Premières étapes de l'algorithme de Guibas et Oudot 85
4.9 Diagramme d'évolution des nombres de Betti 86
4.10 Pour la preuve du lemme 4.14 86
4.11 Complexe de témoins sur une surface 89

5.1 Point location: méthode du slab 94
5.2 Triangulations hiérarchiques 95
5.3 Range search en dimension 1 97
5.4 Range Trees 98
5.5 Fractional cascading 99
5.6 *kd*-trees 101

6.1 Étapes de l'algorithme de Christofides 109
6.2 Étapes de l'algorithme d'Arora 111
6.3 Le quadtree construit par l'algorithme d'Arora 112
6.4 Pour la preuve du lemme 6.7 113
6.5 La table de requêtes construite par l'algorithme d'Arora 115

Première partie

Les fondamentaux de la géométrie algorithmique

Chapitre 1

Géométrie discrète et algorithmique : introduction

1.1 Quelques notions sur les graphes, cartes et maillages

Ici nous essayons de fournir des notions précises et de clarifier les différences et points communs entre graphes, cartes et maillages, compte tenu des thèmes et domaines divers abordés dans cet ouvrage. Nous proposons également des algorithmes et des structures de données qui, dans la mesure du posible, peuvent s'appliquer indistinctement aux trois types d'objets mentionnés ci-dessus, ou tout du moins aux cartes et maillages qui, du point de vue combinatoire et en faisant abstraction des propriétés géométriques, peuvent représenter différents aspects d'un même objet.

1.1.1 Graphes

Définition 1.1. *Un* graphe $G = (V, E)$ *est une paire constituée de :*
- *un ensemble de* sommets $V = (v_1, \ldots, v_n)$
- *une famille* $E = (e_1, \ldots, e_m)$ *d'éléments du produit cartésien* $V \times V = \{(u, v) \mid u \in V, v \in V\}$ *appelés* arêtes.

Une *boucle* est une arête dont les extrémités coïncident ($u = v$). Si une arête apparaît plusieurs fois dans E alors il s'agit d'une *arête multiple*. Un graphe est dit *simple* s'il ne contient pas d'arêtes multiples ni de boucles. Un graphe est *k-connexe* s'il faut supprimer au moins k sommets pour qu'il ne soit plus connexe.

1.1.2 Cartes et subdivisions planaires

Bien que déjà intéressants d'un point de vue algorithmique, les graphes ne suffisent pas à capturer les propriétés d'objets communs en géométrie. Par exemple, dans le cas des maillages, il est usuel de parler de "faces", notion qui n'apparaît pas dans la définition de graphe donnée précédemment.

Définition 1.2. *Étant donné un graphe G, un* dessin planaire *est un plongement cellulaire de G dans $\mathbb{R}^2$, qui satisfait les conditions suivantes :*

(i) *les sommets du graphe sont représentés par des points ;*

(ii) *les arêtes sont représentées par des arcs de courbes ne se coupant qu'aux sommets ;*

(iii) *le complément $\mathcal{S} \setminus \mathcal{G}$ est une union disjointe de régions appelées* faces, *ne contenant ni sommets ni arêtes, qui ont toutes sauf une la topologie d'un disque ouvert, la dernière (appelée* face externe *ou* face infinie*) ayant la topologie d'un disque ouvert avec un trou.*

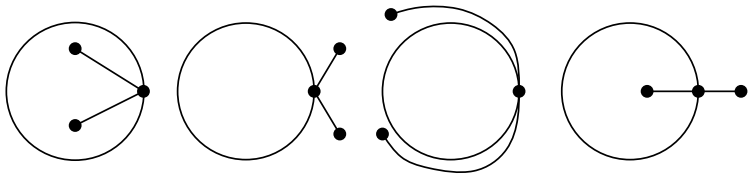


FIGURE 1.1 – *Quatre plongements d'un même graphe planaire sont dessinés ici, définissant deux cartes différentes sur la sphère (3 dans le plan). Les trois premiers plongements, ayant des faces de degré* 1 *et* 5*, représentent la même carte sur la sphère : les deux premiers diffèrent seulement pour le choix de la face infinie (ce qui les rend différents combinatoirement dans le plan), tandis que le troisième peut s'obtenir du deuxième par déformation continue. Le quatrième dessin correspond à une carte différente ayant deux faces de degré* 3*.*

Remarquons que la condition (iii) implique que le graphe associé à une carte planaire doit être connexe, car sinon il existe des faces qui ne sont pas des disques topologiques (ou pas un disque à un trou dans le cas de la face externe).

Un graphe G est dit *planaire* s'il admet un dessin planaire, et dans ce cas on parle de *graphe plan* si l'on considère un dessin planaire particulier de G. Observons qu'un graphe planaire peut admettre plusieurs dessins différents qui ne sont pas équivalents du point de vue de la combinatoire du complémentaire (condition (iii)), comme illustré dans la Figure 1.1.

On peut également voir les graphes planaires d'une autre façon : comme ceux qui peuvent être plongés (dessinés) sur une sphère. Pour cela on peut considérer la *projection stéréographique* Π qui réalise une bijection entre le plan $\mathbb{R}^2$ et une sphère privée d'un point. Considérerons une sphère dans $\mathbb{R}^3$ tangente au plan $z = 0$, et la projection stéréographique à partir du pôle nord N de la sphère. Il est facile de voir qu'étant donné un plongement d'un graphe G sur la sphère tel que le point N n'appartient à aucun arc de G, on obtient un plongement de G dans le plan, qui est bien planaire car sans croisements. De manière similaire, l'application inverse de la projection Π réalise l'autre sens de la bijection. La différence entre le plongement dans le plan et celui sur la sphère est que la face infinie (celle qui passe par le pôle nord N) a la topologie d'un disque, comme les autres faces, tandis qu'elle est semblable à un disque privé d'un point dans le plan. Cette subtilité rend les deux premiers dessins de la figure 1.1 équiavlents sur la sphère mais pas dans le plan.

Si l'on fait abstraction de la géométrie du dessin (les coordonnées des sommets et les arcs de courbe représentant les arêtes), on peut considérer les plongements à homéomorphisme ambiant près, i.e. à transformation bijective et bi-continue près du plan sur lui-même. Dans ce cas on parle de *cartes planaires* (et plus généralement de *cartes topologiques*), définies comme étant les classes d'équivalences de plongements pour l'action des homéomorphismes ambiants. Il existe aussi une autre notion de carte, purement combinatoire, que l'on détaillera au chapitre 3.

Nous introduisons maintenant un concept plus général que les cartes planaires : celui des *subdivision planaires*, définies comme les partitions de $\mathbb{R}^2$ en régions (parfois appelées *cellules*) induites par le plongement d'un ou plusieurs graphes planaires. Toute carte planaire est automatiquement une subdivision planaire. La réciproque est fausse toutefois : premièrement, dans une subdivision il peut y avoir des cellules qui ne sont pas simplement connexes (le graphe correspondant à la subdivision peut ne pas être connexe) ; deuxièmement, certaines cellules d'une subdivision peuvent avoir des arêtes de longueur infinie, comme illustré dans la figure 1.2.

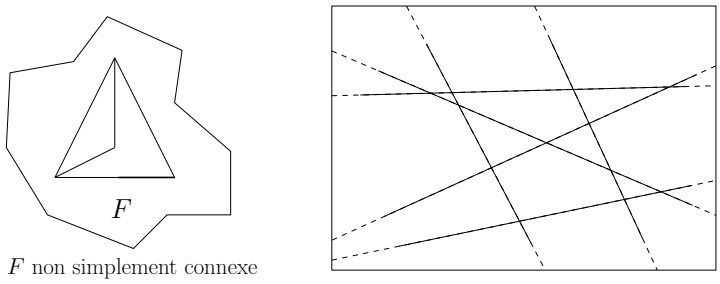


Figure 1.2 – *Exemples de subdivisions planaires qui ne correspondent pas à un graphe planaire.*

1.1.3 Arbres

Un *arbre* est par définition un graphe connexe sans cycles. Ou encore, suivant la terminologie des cartes, un *arbre plan* est une carte planaire ayant une seule face, la face externe. Un arbre à n sommets a toujours $n-1$ arêtes. Pour s'en convaincre, il suffit de choisir un sommet v comme racine et d'orienter toutes les arêtes dans la direction de v le long de l'arbre (cette direction est unique) : ainsi il y a une correspondance bijective entre les arêtes et les sommets dont elles sont issues, qui sont précisément tous les sommets de l'arbre excepté v.

Un arbre *binaire* est un arbre plan tel que tout nœud a 3 voisins, et donc deux fils s'il est enraciné. Un *arbre couvrant* d'un graphe $\mathcal{G}$ est un sous-graphe (connexe et sans cycles) de $\mathcal{G}$ reliant tous ses sommets. Les arbres couvrants joueront un rôle crucial dans de nombreux algorithmes présentés dans ce cours.

1.1.4 Triangulations

Nous allons maintenant parler de *triangulations*, qui constituent l'un des objets fondamentaux de la géométrie algorithmique.

Définition 1.3. *Étant donné un ensemble $P = \{p_1, \ldots, p_n\}$ de points dans $\mathbb{R}^2$, une* triangulation des points de l'ensemble P *est une subdivision planaire maximale dont les sommets correspondent aux points de P. En d'autres termes, il est impossible d'ajouter de nouvelles arêtes joignant des points de P sans introduire des croisements.*

Observons que, en faisant abstraction du plongement géométrique, une *triangulation planaire* est une carte planaire où toutes les faces hormis éventuellement la face externe ont degré 3, chacune étant incidente à trois arêtes. Cette définition se est limite au cas de points situés dans un plan ou sur une sphère, nous la généraliserons aux dimensions supérieures au chapitre 2.

1.1.5 Complexes et maillages

Un *simplexe* dans le plan (respectivement dans $\mathbb{R}^d$) est l'enveloppe convexe de 3 (respectivement $d+1$) points affinement indépendants. Un *complexe simplicial* C dans le plan ou plus généralement dans $\mathbb{R}^d$ est un ensemble de simplexes vérifiant les deux conditions suivantes :

(i) toute face d'un simplexe de C est aussi un simplexe de C ;
(ii) l'intersection de deux simplexes est vide ou bien est constituée d'un simplexe de dimension inférieure (face commune de dimension maximale).

On utilise parfois le terme de *complexe simplicial plongé* pour désigner de telles structures, par opposition aux complexes simpliciaux abstraits qui se définissent de manière purement combinatoire et font abstraction d'une quelconque réalisation géométrique dans $\mathbb{R}^d$ ou tout autre espace ambiant.

C est un *k-complexe* si la dimension maximale des simplexes qui le composent est k, et dans ce cas il est *pur* lorsque toute face est l'une des faces d'un k-simplexe de C. Le *l-squelette* d'un complexe C est le sous-complexe constitué par ses faces de dimension au plus l : le 1-squelette d'un complexe est donc isomorphe à un graphe ayant pour nœuds et arêtes respectivement les faces de dimension 0 et 1 de C.

Plus généralement, on peut définir des complexes dont les k-faces ne sont pas des k-simplexes mais des *cellules* de dimension intrinsèque k. On parle alors de *complexe cellulaire plongé*. La définition formelle est un peu technique car une k-face n'est plus simplement l'enveloppe convexe de $k+1$ sommets dans $\mathbb{R}^d$, mais l'image dans $\mathbb{R}^d$ de la k-boule unité par un homéomorphisme. Pour le reste, les relations d'incidences sont similaires (excepté que le nombre de $(k-1)$-faces incidentes à une k-face peut être supérieur à $k+1$), et comme nous n'aurons pas besoin de la définition précise nous renvoyons simplement le lecteur à [LW69] pour de plus amples détails.

Définition 1.4. *Un* maillage *est un complexe cellulaire plongé dans $\mathbb{R}^d$. Il est dit*

- simplicial *si ses faces sont des simplexes (lui-même est alors un complexe simplicial plongé).*
- surfacique*, ou* surface combinatoire*, s'il est homéomorphe à une surface sans bord. Du point de vue combinatoire, ceci équivaut à dire que le maillage est 2-dimensionnel, que chaque arête est incidente à deux 2-faces exactement, et que chaque sommet a exactement un cycle d'arêtes incidentes.*
- volumique *s'il est homéomorphe à un compact de $\mathbb{R}^3$ dont le bord est une surface.*

Notez que le deuxième item peut être étendu au cas des maillages homéomorphes à des surfaces à bords. Dans ce cas chaque arête est incidente à une ou deux 2-faces, et chaque sommet admet un cycle ou une chaîne d'arêtes incidentes.

L'information combinatoire décrite par les relations d'incidence entre sommets, arêtes et faces de plus grandes dimensions dans un maillage est appelée la *connectivité* du maillage.

Les définitions ci-dessus sont illustrées dans la figure 1.3.

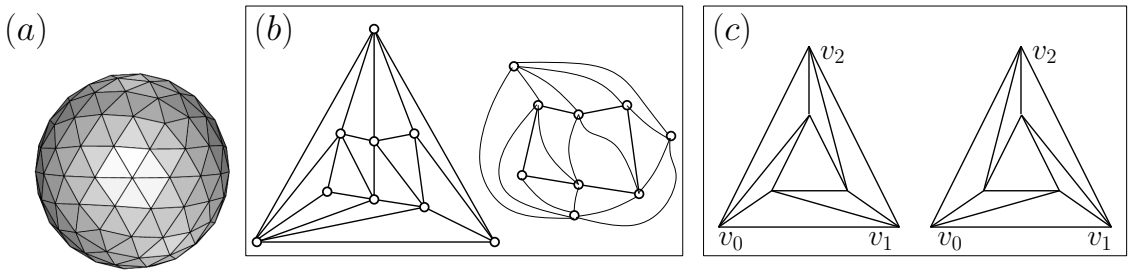


FIGURE 1.3 – *Quelques exemples de maillages. Un maillage surfacique triangulaire (simplicial) (a). Une triangulation d'un ensemble de points dans le plan, et une carte planaire triangulée équivalente (b). Deux triangulations différentes d'un même ensemble de points dans le plan (c) : en tant que cartes, les deux triangulations sont équivalentes.*

1.2 Formule d'Euler en dimension 2

Une fois introduits les concepts de graphe (planaire), carte et maillage, il ne reste qu'à mentionner une propriété fondamentale, introduite par Euler, caractérisant ces trois types d'objets.

Cette relation, admettant une preuve simple dans le cas de surfaces de dimension 2, exprime essentiellement une dépendance linéaire entre le nombre de sommets, arêtes et faces d'un complexe ou d'une carte (n'oublions pas que le 1-squelette d'un 2-complexe est un graphe planaire).

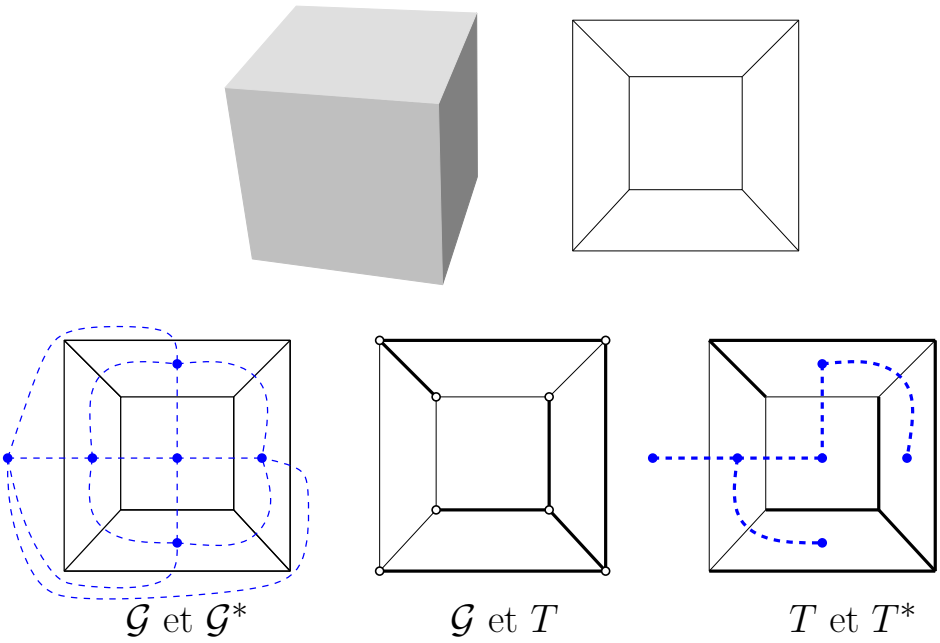


Figure 1.4 – *Ces images illustrent quelques étapes de la preuve de la formule d'Euler.*

Parmi les formulations équivalentes existantes, nous préférons fournir ici celle donnée pour les maillages sphériques, i.e. les maillages surfaciques homéomorphes à une sphère, dont le 1-squelette définit ainsi une carte planaire.

Théorème 1.5 (Formule d'Euler)**.** *Pour tout maillage sphérique (ou carte planaire) ayant n sommets, e arêtes et f faces nous avons :*

$$n - e + f = 2$$

Démonstration. On commence par observer qu'on peut ramener le maillage à une carte planaire : pour cela il suffit de supprimer une face et de déformer de manière continue les arêtes et faces restantes de façon à ce que tous les sommets soient coplanaires. On définit ainsi une carte planaire avec e arêtes, n sommets et $f - 1$ faces.

Soit $\mathcal{G}$ ce graphe planaire auquel on a ajouté la face infinie, et considérons un arbre T couvrant $\mathcal{G}$. Par définition, T est un graphe connexe sans cycle, ayant n sommets et $n - 1$ arêtes. Remarquons que T peut aussi se voir comme une carte planaire à n sommets ayant une seule face (la face infinie) ; il est alors facile de vérifier que pour T la relation d'Euler est satisfaite : $n - (n - 1) + 1 = 2$.

Considérons maintenant le graphe dual $\mathcal{G}^* = (V^*, E^*)$ de $\mathcal{G}$, qui a pour sommet les faces de $\mathcal{G}$ et dont les arêtes correspondent aux paires de faces adjacentes dans $\mathcal{G}$. On va construire un arbre couvrant spécial T^* consistant de toute les arêtes de $\mathcal{G}^*$ qui correspondent aux arêtes de $\mathcal{G} \setminus T$ (dans le primal). Il est facile de voir que T^* couvre tous les sommets de $\mathcal{G}^*$ et est connexe (car T n'a pas de cycle). En plus il s'agit bien d'un arbre, car il n'y a pas de cycles : autrement un tel cycle devrait séparer un deux ensembles (non adjacents) les sommets de T, qui est connexe. Or, T^* est un arbre à f sommets, ayant donc $f - 1$ arêtes et une seule face.

En comptant les nombre de sommets et d'arêtes des deux arbres T et T^* (rappelons que $e = E(T) + E(T^*)$) on arrive enfin à la relation d'Euler : $n + f = (E(T) + 1) + (E(T^*) + 1) = e + 2$. □

Exercice 1.6. *Fournir une autre preuve directe (ne faisant pas intervenir d'arbres couvrants) de la formule d'Euler.*

On peut tout de suite remarquer que, étant donné un graphe planaire $\mathcal{G}$, le nombre de faces est toujours le même et ne dépend pas du plongement planaire choisi pour $\mathcal{G}$, car il est déterminé une fois les nombres d'arêtes et sommets fixés. De plus, la formule d'Euler reste valide lorsque l'on considère des graphes qui peuvent avoir des boucles et arêtes multiples.

Quelques inégalités utiles. Dans le cas particulier de graphes (ou cartes) dont les faces ont degré au moins 3, la relation d'Euler garantit notamment que le nombre d'arêtes est linéairement borné par le nombre de sommets.

De plus, on peut montrer que dans tout graphe planaire il existe des sommets de faible degré : cette remarque, ainsi que le fait que les graphes planaires soient creux, sont deux propriétés largement utilisées en géométrie algorithmique pour obtenir des bornes sur la complexité combinatoire des objets et les performances des algorithmes qui les calculent.

Corollaire 1.7. *Soit G un graphe planaire simple (sans boucles ni arêtes multiples) ayant $n \geq$ sommets et e arêtes. Alors les relations suivantes sont satisfaites :*

- *il existe toujours un sommet de G ayant degré au plus 5.*
- *$e \leq 3n - 6$. De plus, $e = 3n - 6$ si et seulement si G est une triangulation.*

Démonstration. La preuve est basée sur un argument de double comptage. Si on dénote par n_i et f_i le nombre de sommets de degré i et le nombre de faces de taille i respectivement, on peut écrire :

$$f = f_1 + f_2 + f_3 + \dots$$
$$n = n_1 + n_2 + n_3 + \dots$$

Or, étant donné que toutes les faces ont degré au moins 3 (car $\mathcal{G}$ est simple), cela peut s'écrire $f = f_3 + f_4 + \dots$.

Maintenant il ne reste qu'à compter les arêtes contenues dans chaque face : comme chaque arête apparaît deux fois (dans la même face éventuellement), on obtient :

$$2e = 3 \cdot f_3 + 4 \cdot f_4 + \dots$$

d'où la relation $2e - 3f \geq 0$.

Si, par l'absurde, on avait que tout sommet a degré au moins 6 alors on pourrait écrire :

$$n = n_6 + n_7 + n_8 + \dots$$

et avec un double comptage des arêtes incidentes aux sommets :

$$2e = 6 \cdot n_6 + 7 \cdot n_7 + 8 \cdot n_8 + \dots$$

d'où une deuxième relation : $2e - 6n \geq 0$.

Les deux inégalités ci-dessous impliquent que $e - n - f \geq 0$ car

$$6(e - n - f) = (2n - 6) + 2(2e - 3f) \geq 0$$

d'où la contradiction : la formule d'Euler nous dit que $e = n + f - 2$ (au lieu de $e \geq n + f$).

Et pour terminer, comme déjà observé $2e - 3f \geq 0$: ainsi en appliquant la formule d'Euler on trouve

$$3n - 6 = 3(e - f + 2) = 3e - 3f \geq 0$$

ce qui prouve la deuxième partie de l'énoncé. □

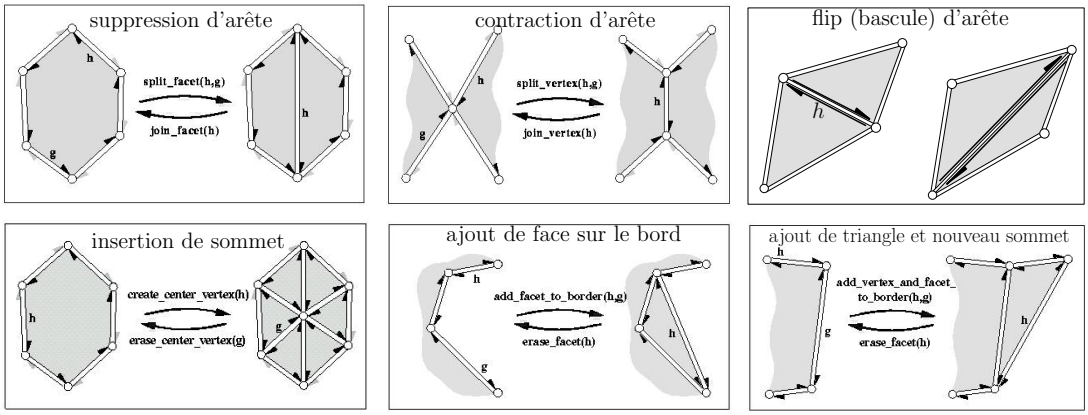


Figure 1.5 – *Opérations d'Euler permettant de modifier un maillage correspondant à une 2-variété orientable, de genre arbitraire et sans ou avec bords. Les opérations illustrées ici modifient la structure combinatoire du maillage sans affecter la géométrie des sommets. (certaines de ces images proviennent de la documentation de la librairie CGAL,* `www.cgal.org`)

1.3 Représentations de données géométriques

1.3.1 Structures de données abstraites pour la géométrie

Une *structure de données abstraites (SDA)* est définie par un ensemble de données, qui est muni d'une collection d'opérations (souvent appelé l'*interface*) permettant la mise à jour et l'accès aux données.

Dans notre cas les données représentent l'information géométrique (par exemples les coordonnées de points dans $\mathbb{R}^d$) ainsi que l'information combinatoire (par exemple, décrivant les relations d'incidences dans un maillage). Et en ce qui concerne l'interface, beaucoup dépend des objets dont on dispose et des manipulations à effectuer.

Pour les cartes et les maillages, qui sont les objets principaux qu'on traite en géométrie algorithmique, on suppose disposer souvent des méthodes suivantes pour la navigation :

— `adjacent`(f, e) : renvoie la face adjacente à f et incidente à l'arête commune e.
— `next_neighbor`(v, w) : étant donné une arête (v, w) renvoie le prochain sommet incident à v, à partir de w (par exemple en sens horaire).
— `adjacent`(v, w) : teste si deux sommets sont adjacents.

La Figure 1.5 illustre quelques opérations standard permettant la modification de la structure combinatoire d'un maillage [1] :

1.3.2 Quelques structures de données pour les maillages

Étant donnés les nombreux domaines d'applications où l'on utilise les maillages, plusieurs structures de données ont été conçues fournissant une implementation de l'interface décrite ci-dessus. Nous en présentons quelques unes dans le reste de cette section.

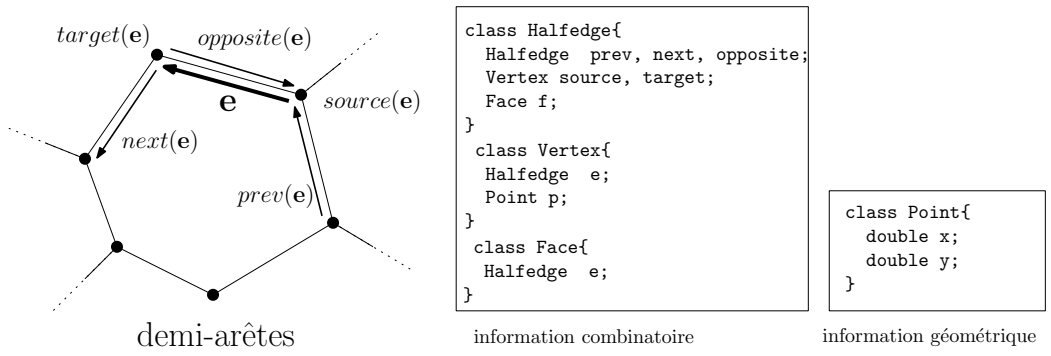


FIGURE 1.6 – *Représentation d'une carte à l'aide de demi-arêtes.*

1.3.2.1 Maillages surfaciques (polygonaux) : Half-edge

La structure de demi-arêtes est une représentation de la combinatoire des cartes planaires, des polyèdres et, plus généralement, de toute *surface polygonale orientable de dimension* 2, plongée dans un espace $\mathbb{R}^d$ de dimension quelconque (une implementation de cette structure est décrite dans [Ket99]). Chaque arête est décomposée en deux demi-arêtes, ayant orientations opposées, étant incidente à un sommet source et à un sommet destination. L'élément de base est la demi-arête (*half-edge*) qui est représentée à l'aide des références suivantes : une référence à la demi-arête opposée, ainsi qu'une autre vers la demi-arête qui suit dans la même face ; on a aussi une référence à la face incidente, ainsi qu'aux sommets source et destination (on peut éventuellement aussi stoquer la demi-arête qui précède en sens horaire, dans la même face incidente). Et pour terminer, chaque face et sommet est représenté en stoquant une référence vers l'une des demi-arêtes incidentes. Cette structure de données est illustrées par la Figure 1.6.

1.3.2.2 Une structure de données pour les triangulations

Lorsque on doit manipuler des maillages triangulaires, il est parfois plus convenable d'utiliser une implementation différente [BDP+02], qui peut se révéler bien moins gourmande en espace mémoire par rapport aux représentations décrites auparavant. Une solution simple consiste à utiliser une représentation où l'élément principal est le triangle : il devient donc plus nécessaire de stoquer explicitement les arêtes. Chaque triangle contient trois références vers ses faces voisines, ainsi que 3 références vers ses sommets incidents (voir la Figure 1.7). On représente un sommet avec une référence vers l'un de ses triangles incidents (et comme toujours, on stoque une références vers ses coordonnées géométriques).

Il est facile de vérifier qu'avec une telle structure on peut implémenter toutes les méthodes de l'interface présentée ci-dessus.

Un avantage de cette structure est le fait que, avec quelques simples modifications, il est possible de représenter des maillages triangulaire volumiques (maillages tétraédriques).

1.3.2.3 Quad-edge data structure

Une autre structure de données basées sur la représentation des arêtes comme objet de base est celle proposée par Guibas et Stolfi [GS85], surnommée *Quad-Edge*. L'avantage principale réside dans la possibilité de représenter le graphe primal ainsi que son dual au même temps : ce

1. Les maillages sont supposés correspondre à des 2-variétés orientables, avec ou sans bord : le voisinage de chaque sommet est homéomorphe à un disque topologique (ou éventuellement à un demi-disque en présence de bords).

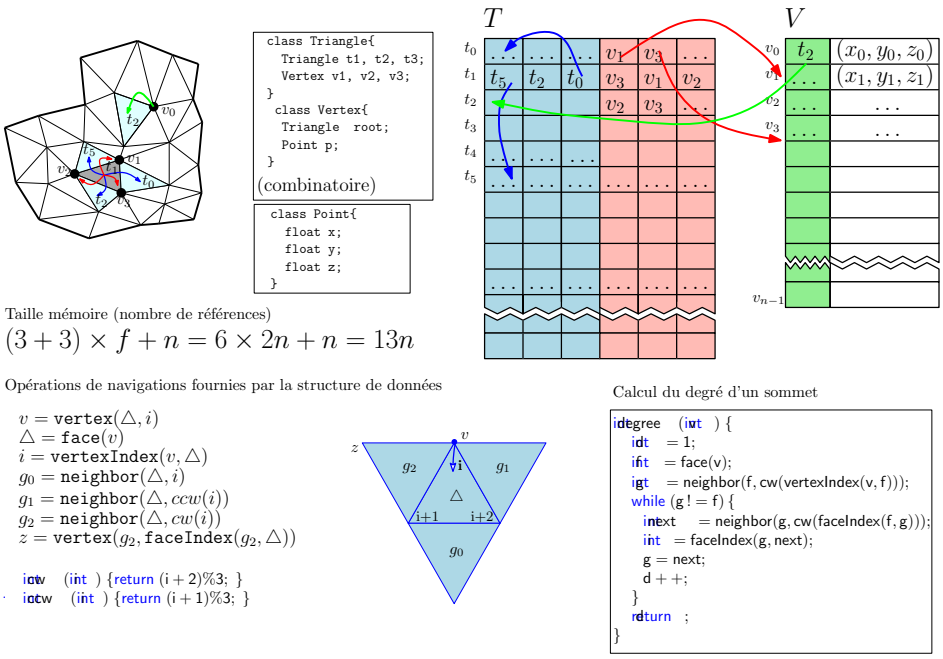


FIGURE 1.7 – *Représentation d'un maillage triangulaire à base de triangles.*

qui se révèle assez pratique, par exemple, dans la manipulation de la triangulation de Delaunay et de son dual, le diagramme de Voronoi.

L'objet de base est l'arête, qui est représentée par quatre *quad-edges*, dénotés $e[0], e[1], e[2], e[3]$: $e[0]$ et $e[2]$ correspondent aux deux orientations de l'arête e, alors que $e[1], e[3]$ représentent les deux version orientées de l'arête e^* (la duale de e).

1.4 Un problème classique : localisation planaire

Comme dans d'autres domaines de l'informatiques il existe quelques problèmes fondamentaux, suggérés souvent par les applications : souvent leur solution a motivé un certain nombre de structures de données et algorithmes de la géométrie algorithmique, dont une partie sera traitée dans cet ouvrage.

L'un des problèmes les plus étudiés dès les débuts de la Géométrie Algorithmique est celui de la localisation de points dans une subdivision planaire.

Le but est de faire un prétraitement des données (en particulier de la combinatoire de la carte planaire qui décrit la subdivision) afin de permettre une implementation rapide de certaines de requêtes de localisation : le plus souvent, pour une subdivision de taille n on vise un temps de requête $O(\log n)$, qui est optimal si on se place dans le modèle de calcul basé sur les comparaisons. Il y a aussi d'autres critères dont tenir compte : par exemple, la quantité d'espace utilisé par la structure de données (qu'on cherche de taille $O(n)$), ainsi que le complexité en temps de la phase de prétraitement.

Problème 1.8 (Planar point location)**.** *Étant donné une subdivision du plan en cellules (donnée par une carte), déterminer la cellule qui contient un point donné p.*

Straight walk algorithm. Si on dispose d'une triangulation d'un ensemble de points qui sont bien repartis dans le plan (par exemple selon une distribution uniforme dans un carré),

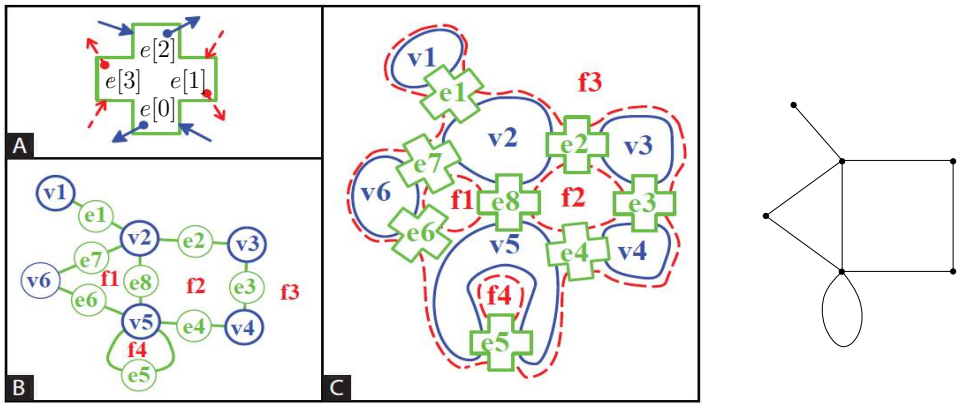


FIGURE 1.8 – *Représentation d'un maillage à l'aide de la structure* Quad-Edge. *Le maillage est représenté à l'aide de trois objets (B) : les sommets, les faces et les* quad-edges. *Chacune des quatre aretes formant un quad-edge contient des champs* suivant *et* précédent *permettant d'effectuer la navigation autour de chaque sommet (A).*
(images de Abdelkrim Mebarki)

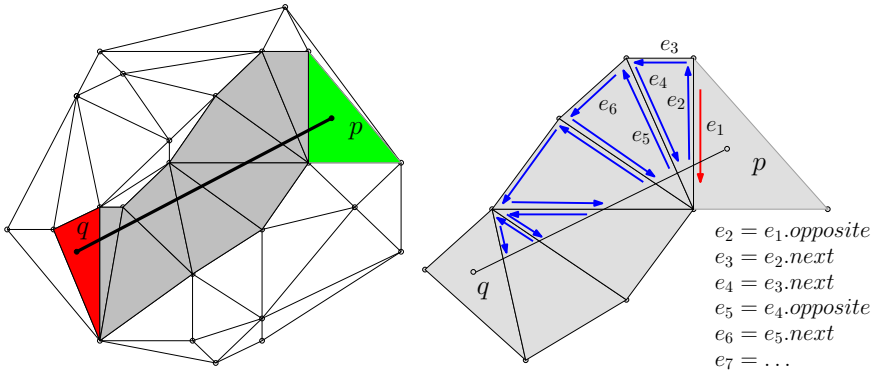


FIGURE 1.9 – *Localisation dans une triangulation planaire.*

alors il existe un algorithme très simple à mettre en place, qui ne nécessite pas d'une phase de prétraitement. Pour localiser le point p (le point de requête), on choisit d'abord un autre point q dans un triangle choisi aléatoirement (cette phase nécessite temps $O(1)$). On considère alors le segment (p, q) et on effectue une *marche* à travers la triangulation, à partir de q jusqu'à p : cela consiste à visiter les triangles qui ont une intersection non nulle avec le segment (p, q). Si on dispose d'une structure de donnée telle que *Quad-edge* (où *half-edge*, où autre similaire) permettant de tester efficacement les relations d'adjacences entre triangles, alors cette marche nécessite un temps $O(k)$ proportionnel au nombre k de triangles à visiter. Il s'avère que pour certaines classes de triangulations (telle que la triangulation de Delaunay), on peut prouver que le temps de calcul pour la localisation est $O(\sqrt{n})$ (*expected time*) pour une triangulation de n points [BD98]. Entre autre, cette stratégie permet de calculer une triangulation de Delaunay en temps $O(n\sqrt{n})$ dans le cas de points distribués uniformément.

Ces performances sont bien loin de l'optimal, comme on le verra au Chapitre 5.

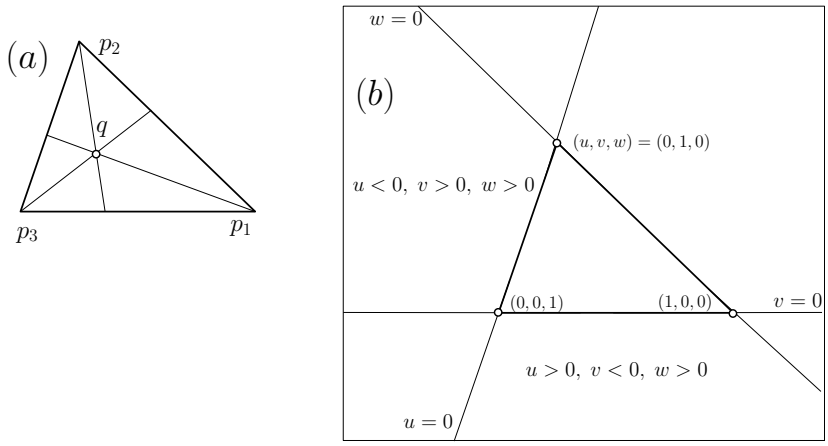


Figure 1.10 – *Système de coordonnées barycentriques définies par un triangle dans le plan.*

1.5 Géométrie convexe

Le but de cette section est de rappeler quelques résultats importants de géométrie convexe qui seront utiles dans la suite.

1.5.1 Coordonnées barycentriques

Étant donné un point q qui peut s'écrire comme *combinaison barycentrique* sous la forme $q = \sum_i^n \alpha_i p_i$ (avec $\sum_i \alpha_i = 1$), on dit que les coefficients $(\alpha_1, \ldots, \alpha_n)$ sont les *coordonnées barycentriques* de q par rapport aux points $p_1, \ldots, p_n$.

Si on considère trois points non colinéaires p_1, p_2 et p_3 dans $\mathbb{R}^2$ il est toujours possible de calculer les coordonnées barycentriques d'un point donné q par rapport à p_1, p_2, p_3 : en effet, cela revient à résoudre un système de 3 équations en trois inconnues. Il existe une jolie interprétation des coordonnées barycentriques dans le plan en termes d'aires normalisées. En appliquant la règle de Cramer pour le calcul de l'aire d'un triangle on trouve (voir Fig. 1.10(a)) :

$$\mathbf{q} = u\,\mathbf{p_1} + v\,\mathbf{p_2} + w\,\mathbf{p_3} = \frac{aire(q, p_2, p_3)}{aire(p_1, p_2, p_3)}\mathbf{p_1} + \frac{aire(p_1, q, p_3)}{aire(p_1, p_2, p_3)}\mathbf{p_2} + \frac{aire(p_1, p_2, q)}{aire(p_1, p_2, p_3)}\mathbf{p_3}$$

Une propriété importante est que les points q pour lesquels tous les trois coefficients barycentriques sont positifs sont ceux exactement à l'intérieur du triangle (p_1, p_2, p_3). Alors que les autres points, à l'extérieur, ont au moins un coefficient négatif, comme illustré par la Figure 1.10(b).

1.5.2 Enveloppes affines, enveloppes convexes

Définition 1.9. *Soit $\mathcal{S} = \{p_1, \ldots, p_n\}$ un ensemble fini de points dans $\mathbb{R}^d$.*
• *L'*enveloppe affine *de $\mathcal{S}$ est l'ensemble des combinaisons linéaires de points de $\mathcal{S}$:*

$$\text{aff}(\mathcal{S}) = \{ \sum_{i=1}^{n} \alpha_i p_i \mid \sum_i \alpha_i = 1 \}.$$

• *L'*enveloppe convexe *de $\mathcal{S}$ est l'ensemble des combinaisons linéaires convexes de points de $\mathcal{S}$:*

$$\text{conv}(\mathcal{S}) = \{ \sum_{i=1}^{n} \alpha_i p_i \mid \alpha_i \geq 0, \sum_i \alpha_i = 1 \}.$$

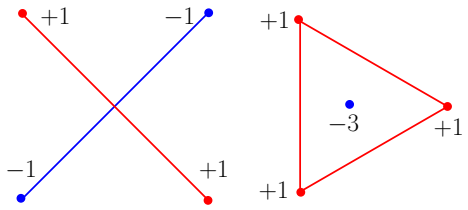


FIGURE 1.11 – *Ces images illustrent le théorème de Radon.*

Un ensemble de points $\mathcal{S}$ est *convexe* si pour toute paire de points $p, q \in \mathcal{S}$ le segment de droite ayant ces deux points comme extrémités est entièrement contenu dans $\mathcal{S}$. Notez que $\text{conv}(\mathcal{S})$ est également le plus petit ensemble convexe contenant $\mathcal{S}$, i.e. c'est l'intersection de tous les convexes contenant $\mathcal{S}$.

1.5.3 Théorèmes fondamentaux : Radon, Helly et Carathéodory

Théorème 1.10 (Radon)**.** *Étant donné un ensemble $\mathcal{S} = \{p_1, \ldots, p_n\}$ de points de $\mathbb{R}^d$ avec $n \geq d+2$, $\mathcal{S}$ peut être partitionné en deux sous-ensembles $\mathcal{S}_1$ et $\mathcal{S}_2$ tels que*

$$\text{conv}(\mathcal{S}_1) \bigcap \text{conv}(\mathcal{S}_2) \neq \emptyset$$

Démonstration. Considérons $d+2$ points $\{p_1, \ldots, p_{d+2}\} \subset S$: puisque on est en dimension d, ces points ne peuvent pas etre affinement indépendants. Ce qui revient à dire qu'un point, disons p_{d+2}, peut s'exprimer comme combinaison affine des autres :

$$p_{d+2} = \sum_{i=1}^{d+1} \alpha_i p_i, \text{ où } \sum_{i=1}^{d+1} \alpha_i = 1$$

Il existe donc $d+2$ coefficients $\alpha_1, \ldots \alpha_{d+2}$ (qui ne sont pas tous nuls), satisfaisant les $d+1$ équations suivantes :

$$\sum_{i=1}^{d+2} \alpha_i = 0 \quad \text{et} \quad \begin{cases} \sum_{i=1}^{d+2} \alpha_i p_i^x = 0 & (p_i^x \text{ étant la coordonnée } x \text{ de } p_i) \\ \sum_{i=1}^{d+2} \alpha_i p_i^y = 0 \\ \ldots \end{cases} \tag{1.1}$$

Dénotons par P et N les ensembles d'indices i correspondants aux coefficients α_i qui sont positif (ou nuls) et négatifs respectivement. On va montrer que P et N définissent la partition $\mathcal{S}_1, \mathcal{S}_2$ recherchée. Pour cela il suffit de considérer le point q donné par

$$q = \sum_{i \in P} \frac{\alpha_i}{A} p_i, \text{ où on définit } A = \sum_{i \in P} \alpha_i \tag{1.2}$$

qui doit appartenir à un convexe $\text{conv}(S_1)$ (enveloppe convexe des points p_i ayant indice $i \in P$), puisque q est une combinaison convexe, car $\sum_{i \in P} \frac{\alpha_i}{A} = 1$ par construction. Mais d'un autre coté, à cause des équations 1.1 le point q peut s'exprimer également comme combinaison convexe des points p_i ayant indice dans N :

$$q = \sum_{i \in N} \frac{-\alpha_i}{A} p_i, \text{ avec cette fois } A = \sum_{i \in N} -\alpha_i \tag{1.3}$$

On a ainsi terminé la preuve du théorème de Radon, car on a trouvé un point q qui appartient aux deux enveloppes convexes de $\mathcal{S}_1$ et $\mathcal{S}_2$ (qui sont, par définition des ensembles P et N, disjointes). □

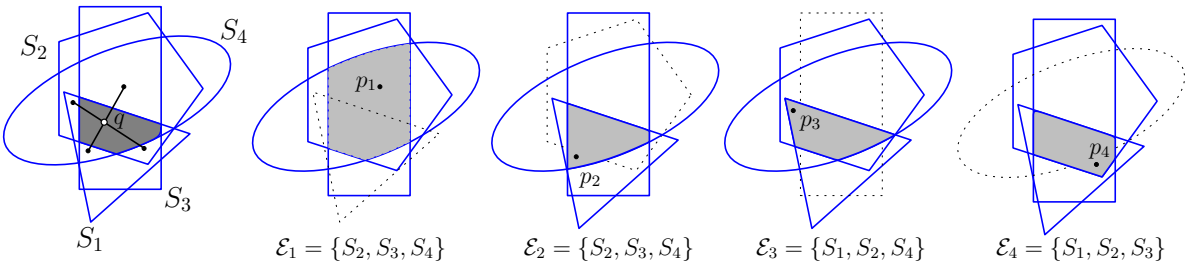


Figure 1.12 – *Ces images illustrent le théorème de Helly (pour $n = 4$ et $d = 2$).*

La preuve du théorème de Radon est constructive, au sens où elle fournit une manière de calculer un *point de Radon* (un point dans $\text{conv}(\mathcal{S}_1) \bigcap \text{conv}(\mathcal{S}_2)$) : pour cela il suffit de déterminer les $d+2$ coefficients α_i solution des équations 1.1. Les points de Radon jouent un role crucial dans la conception d'algorithmes probabilistes pour le calcul approché de *center points* (dont la définition est donné plus loin dans cette section), comme on le détaillera au Chap. 6

Pour l'instant, on peut déjà tester l'utilité du théorème de Radon en donnant la preuve du théorème de Helly, l'un des résultats fondamentaux de géométrie convexe.

Théorème 1.11 (Helly, 1913)**.** *Considérons une famille finie $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ de convexes de $\mathbb{R}^d$ (avec $n \geq d+1$). Si l'intersection de toute sous-famille de taille $d+1$ est non vide alors on a aussi*

$$\bigcap_{i=1}^{n} \mathcal{S}_i \neq \emptyset$$

Démonstration. Pour démontrer ce théorème on procède par induction sur n. Pour $n = d+1$, le cas de base, l'énoncé est vrai car il y a une seule famille taille $d+1$. Supposons que l'énoncé est vrai pour $n-1$, et considérons une collection de n convexes $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$. Soit $\mathcal{E}_i = \{\mathcal{S}_j\}_{j \neq i}$ une collection de $n-1$ convexes ($1 \leq i \leq n$). Par hypothèse d'induction, les convexes de la famille $\mathcal{E}_i$ ont une intersection non vide, donc il existe un point $p_i \in \mathcal{S}_j$ ($\mathcal{S}_j \in \mathcal{E}_i$). Mais si on définit l'ensemble $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, pour le théorème de Radon il existe une partition P_1, P_2, telle que $\text{conv}(P_1) \cap \text{conv}(P_2)$ n'est pas vide, et contient un certain point q. Supposons que, étant donné un convexe $\mathcal{S}_i$, le point p_i appartient à l'ensemble P_1. Observons que un point $p_j \in C$, pour tout convexe C de la collection $\mathcal{E}_j$, et que $C_i \in \mathcal{E}_j$ lorsque $j \neq i$.

Mais alors $P_2 \subset C_i$, puisque pour tous les points $p_j \in P_2$ on a $p_j \in \mathcal{S}_i$. On peut donc déduire que $q \in \mathcal{S}_i$, puisque par définition $q \in \text{conv}(P_2)$, et $\text{conv}(P_2) \subset \mathcal{S}_i$ (car $\mathcal{S}_i$ est convexe par hypothèse). On peut donc conclure que l'énoncé du théorème est vrai pour toute famille de convexes de taille n, car on vient de montrer que $q \in \mathcal{S}_i$ pour tout $1 \leq i \leq n$. □

Un autre résultat fondamental de géométrie convexe est le théorème de Carathéodory, qui dit que si un point p est contenu dans l'enveloppe convexe d'un certain ensemble $\mathcal{S}$ en dimension d, alors il est aussi contenu dans un $r-$simplexe (avec $r \leq d$) dont les extrémités appartiennent à $\mathcal{S}$.

Théorème 1.12 (Carathéodory, 1911)**.** *Étant donné un ensemble $\mathcal{S} = \{p_1, \ldots, p_n\}$ de points de $\mathbb{R}^d$, tout point $q \in \text{conv}(\mathcal{S})$ peut s'exprimer comme combinaison convexe d'au plus $d+1$ points de $\mathcal{S}$.*

Démonstration. Puisque $q \in \text{conv}(\mathcal{S})$ alors q peut s'exprimer comme combinaison convexe des points de $\mathcal{S}$. Supposons, par absurde, que r (avec $r > d+1$) est la taille du plus petit sous-

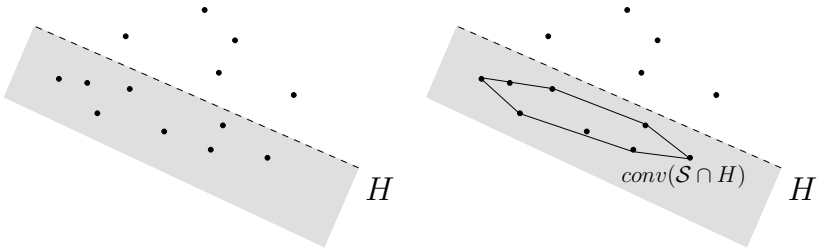


FIGURE 1.13 – *Ces images illustrent le théorème du center point.*

ensemble $S' = \{p'_1, \ldots, p'_r\}$ de $\mathcal{S}$, tel que

$$q = \sum_{i=1}^{r} \alpha_i p'_i \text{ , où } \sum_i \alpha_i = 1 \text{ et } \alpha_i \geq 0.$$

Puisque on est en dimension d, et $r > d+1$, les points $p'_1, \ldots, p'_r$ ne peuvent pas etre indépendants : comme déjà observé pour le théorème de Radon, il existe alors des coefficients $\beta_1, \ldots, \beta_r$ (qui ne sont pas tous nuls) tels que :

$$\sum_{i=1}^{r} \beta_i p'_i = 0 \text{ , et } \sum_i \beta_i = 0$$

Supposons, sans perte de généralité, que $\beta_r > 0$ et $\frac{\alpha_r}{\beta_r} \leq \frac{\alpha_i}{\beta_i}$, pour tous les $i < r$ tels que $\beta_i > 0$.

Définissons maintenant $r-1$ coefficients $\gamma_1, \ldots, \gamma_{r-1}$ de la manière suivante :

$$\gamma_i = \alpha_i - \frac{\alpha_r}{\beta_r}\beta_i$$

qui sont tous non négatifs par définition, car

$$\begin{cases} \gamma_i \geq \alpha_i \geq 0 & \text{si } \beta_i \leq 0 \\ \gamma_i = \alpha_i - \frac{\alpha_r}{\beta_r}\beta_i \geq \alpha_i - \frac{\alpha_i}{\beta_i}\beta_i = 0 & \text{si } \beta_i > 0 \end{cases}$$

et de plus forment une partition de l'unité :

$$\sum_{i=1}^{r-1} \gamma_i = \sum_{i=1}^{r-1} (\alpha_i - \frac{\alpha_r}{\beta_r}\beta_i) = \sum_{i=1}^{r-1} \alpha_i - \frac{\alpha_r}{\beta_r} \sum_{i=1}^{r-1} \beta_i = \sum_{i=1}^{r} \alpha_i - \frac{\alpha_r}{\beta_r} \sum_{i=1}^{r} \beta_i = \sum_{i=1}^{r} \alpha_i = 1$$

Mais alors on a atteint une contradiction, car on peut exprimer le point q comme une combinaison convexe de $r-1$ points de $\mathcal{S}$:

$$\sum_{i=1}^{r} \alpha_i p'_i = \sum_{i=1}^{r} \alpha_i p'_i - \frac{\alpha_r}{\beta_r}(\sum_{i=1}^{r} \beta_i p'_i) = \sum_{i=1}^{r-1} (\alpha_i - \frac{\alpha_r}{\beta_r}\beta_i) p'_i = \sum_{i=1}^{r-1} \gamma_i p'_i$$

□

1.5.4 Existence de center points

Une jolie application du théorème de Helly est la preuve de l'existence d'un center point pour les nuages de points dans $\mathbb{R}^d$ (voir le théorème 1.14 ci-dessous). En gros, un center point est une généralisation du concept de médian en dimensions supérieures :

Définition 1.13. *Étant donné un ensemble $\mathcal{S} = \{p_1, \ldots, p_n\}$ de points dans $\mathbb{R}^d$, un point $x \in \mathbb{R}^d$ est un* center point *de $\mathcal{S}$ si tout demi-espace fermé contenant x contient au moins $\frac{n}{d+1}$ points de X.*

Théorème 1.14 (Center Point)**.** *Tout ensemble fini $\mathcal{S}$ de points de $\mathbb{R}^d$ admet au moins un center point .*

Démonstration. La preuve fait appel au théorème de Helly de la manière suivante. Soit $\mathcal{H}$ l'ensemble de tous les demi-espaces ouverts contenant strictement plus de $\frac{dn}{d+1}$ points de l'ensemble $\mathcal{S}$. Pour tout demi-espace $H \in \mathcal{H}$ dénotons par P_H l'enveloppe convexe des points de $\mathcal{S}$ contenus dans H : $P_H = \text{conv}(\mathcal{S} \cap H)$. Considérons une famille $\mathcal{F}$ quelconque constituée de $d+1$ demi-espaces $\{H_1, H_2, \ldots, H_{d+1}\} \subseteq \mathcal{H}$. Puisque chaque demi-espace contient plus de $\frac{dn}{d+1}$ points de $\mathcal{S}$, en comptant avec multiplicité les points de $\mathcal{S}$ contenus dans les demi-espaces de $\mathcal{F}$ on arrive à un total strictement supérieur à dn. Ceci implique qu'un point de $\mathcal{S}$ a multiplicité au moins $d+1$ dans l'ensemble de $d+1$ demi-espaces, et donc appartient à tous. Ainsi, les éléments de la famille $\mathcal{F}$ ont une intersection commune non vide, qui contient un point de $\mathcal{S}$. Il s'ensuit que les enveloppes convexes $\{P_{H_i}\}_{1 \leq i \leq d+1}$ ont également une intersection commune non vide. Ceci étant vrai pour toute famille de $d+1$ enveloppes convexes $\{P_{H_1}, \cdots, P_{H_{d+1}} \mid H_1, \cdots, H_{d+1} \in H\}$, on conclut par le théorème de Helly (théorème 1.11) que l'intersection de toutes les enveloppes convexes P_H pour H dans $\mathcal{H}$ est non vide. Tout point x dans cette intersection est un center point de $\mathcal{S}$: en effet, pour tout demi-espace fermé H contenant x, le complémentaire de H dans $\mathbb{R}^d$ est un demi-espace ouvert ne contenant pas x et donc n'appartenant pas à $\mathcal{H}$, ce qui fait que H contient au moins $n - \frac{dn}{d+1} = \frac{n}{d+1}$ points de $\mathcal{S}$. □

En général un ensemble $\mathcal{S}$ peut avoir plus d'un center point et un tel center point ne coïncide pas forcement avec l'un des points de $\mathcal{S}$.

Exercice 1.15 (center point approché)**.** *Montrer que étant donné un ensemble S de $d+2$ points en $\mathbb{R}^d$, on peut calculer en temps $O(d^3)$ un point $c \in \mathbb{R}^d$ tel que tout demi-espace fermé contenant c contient au moins $\frac{\beta n}{d+1}$ points, avec $\beta = \frac{2}{d+2}$. (**Hint :** rappeler la manière dont on peut calculer un point de Radon.)*

1.5.5 Combinatoire des Polytopes

Définition 1.16. *Un* polytope *est l'enveloppe convexe d'un ensemble fini de points dans $\mathbb{R}^d$.*

Un simplexe de dimension k est donc un polytope à $k+1$ sommets affinement indépendants dans $\mathbb{R}^d$, pour $k \leq d$. Un d-polytope est dit *simplicial* si toutes ses faces de dimension au plus $d-1$ sont des simplexes. Notez que cela ne signifie pas que le polytope est un simplexe, puisque la face de dimension d peut ne pas être un simplexe.

Théorème 1.17 (de la borne supérieure [McM70])**.** *Tout polytope à n sommets dans $\mathbb{R}^d$ possède au plus $O(n^{\lfloor d/2 \rfloor})$ faces au total. La borne supérieure est optimale et atteinte entre autres par les* polytopes cycliques*, i.e. ceux dont les sommets sont échantillonnés sur la* courbe des moments *paramétrée par $t \mapsto (t, t^2, t^3, \cdots, t^d)$.*

1.6 Enveloppes convexes en 2D : algorithmes et complexité

Le calcul des enveloppes convexes constitue l'un des problèmes plus importants et étudiés de la géométrie algorithmique dès ses origines : ce qui explique le vaste nombre de stratégies développées dans ce contexte. Le but de cette section est de présenter quelques unes des méthodes plus populaires pour le calcul des enveloppes, et surtout de donner un aperçu d'un

certain nombre de paradigmes algorithmiques qui se révèlent crucial dans plusieurs situations. Comme on le verra, la conception d'une stratégie efficace pour le calcul des enveloppes convexes nécessite à la fois la compréhension du problème géométrique, ainsi que la maîtrise de structures de données et techniques algorithmiques.

Dans toute la suite le nuage de points fourni en entrée est noté $\mathcal{S} = \{p_1, \ldots, p_n\}$.

1.6.1 Algorithme de Jarvis

Le premier algorithme que l'on va considérer a été concu par Jarvis en 1973 [Jar73] et ne demande pas de phase de prétraitement, se révélant assez facile à mettre en place. En gros, l'approche construit une sorte de "paquet cadeau" qui entoure les points de données. À chaque étape de la procédure on sélectionne le prochain point à ajouter à l'enveloppe par un critère de minimisation d'angle.

Le premier point à prendre en compte est le point d'ordonnée minimale $u_1 = (u_{1x}, u_{1y})$: par convention on ajoute aussi le point $u_0 = (-\infty, u_{1y})$, qui sera retiré à la fin du processus de sélection, et qui n'intervient que dans la première étape de construction.

Si l'on note $\mathcal{H} = \{u_1, u_2, \ldots, u_i\}$ l'enveloppe convexe à l'étape i, alors le point p à insérer à l'étape $i+1$ est celui qui appartient à $\mathcal{S} \setminus \{u_2, \ldots, u_i\}$ et qui minimise l'angle $\angle(u_{i-1}u_i, u_ip)$. La procédure termine lorsque le point p à insérer coïncide avec le premier point u_1.

Algorithm 1 Calcul de l'enveloppe convexe : algorithme de Jarvis

```
Entrée : S un ensemble de points.
u_1 = le point le plus bas de S;
u_0 = (−∞, u_1y);
i = 1;
Faire
    si i ≠ 1 alors S = S \ {u_i};
    min = +∞;
    Pour tout w ∈ S
        si ∠(u_{i−1}u_i, u_iw) < min alors
            min = ∠(u_{i−1}u_i, u_iw); u_{i+1} = w;
    i = i + 1;
Tant que u_i ≠ u_1
```

Théorème 1.18. *L'algorithme de Jarvis calcule l'enveloppe convexe de n points en temps $O(nh)$, où h est la taille de la sortie (le nombre de points extrémaux), ce qui donne un temps $O(n^2)$ dans le pire cas.*

Démonstration. La première phase de sélection de u_1 s'exécute clairement en temps linéaire $O(n)$. A chaque étape, la recherche du point p qui minimise l'angle $\angle(u_{i-1}u_i, u_ip)$ nécessite au plus $O(n)$. Étant répétée exactement $h-1$ fois, elle donne une complexité totale de $O(nh)$. Ainsi dans le pire cas, lorsque $h = \Omega(n)$, l'algorithme a une complexité quadratique. □

Bien que le temps de calcul de l'algorithme de Jarvis ne soit pas optimal (comme on le verra 1.6.4), cet algorithme peut s'avérer meilleur par rapport à d'autres stratégies (optimales), lorsque la taille de la sortie est petite (typiquement $h = O(\log n)$).

1.6.2 Approche par balayage : l'algorithme de Graham-Andrew

L'idée du balayage est de construire l'enveloppe convexe incrémentalement en traitant les points du nuage par ordre croissant d'abcisses. Plus précisément, on balaye le nuage horizon-

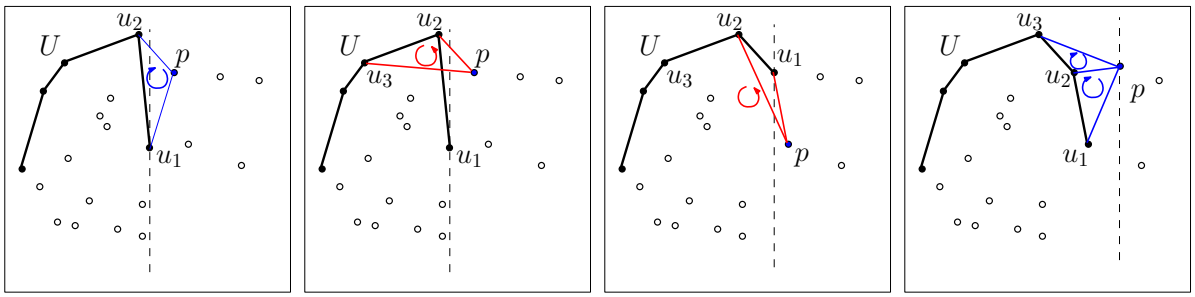


FIGURE 1.14 – *Algorithme de Graham-Andrew pour le calcul de l'enveloppe convexe.*

talement de gauche à droite par une droite verticale, tout en maintenant l'enveloppe convexe des points situés à gauche de la droite. À chaque itération on insère un nouveau point dont on sait qu'il se trouve sur le bord de l'enveloppe convexe des points à gauche de la droite, étant lui-même localisé sur la droite à ce moment-là. Cette stratégie, proposée par Graham [2] [Gra72], permet de calculer l'enveloppe convexe en temps $O(n \log n)$, où n est le nombre de points du nuage. Un autre avantage de cet algorithme est que si les points sont fournis déjà triés par ordre croissant d' abscisses, alors la complexité du calcul est réduite à $O(n)$.

Étant donné un ensemble $\mathcal{S} = \{p_1, \cdots, p_n\}$ de points du plan, l'algorithme commence par trier les points par ordre croissant d'abcisses, de manière à ce qu'on ait (après renommage des points) $p_{1x} \leq p_{2x} \leq \cdots \leq p_{nx}$. Ensuite l'algorithme se divise en deux procédures très similaires permettant de calculer respectivement les enveloppes convexes supérieure et inférieure de $\mathcal{S}$. Une fois ces deux enveloppes connues, il est facile de fusionner les deux listes de sommets afin d'obtenir $\text{conv}(\mathcal{S})$.

La procédure pour le calcul de l'enveloppe supérieure est donnée par le pseudo-code 2. Elle visite les sommets de $\mathcal{S}$ un par un de gauche à droite, en partant de p_1. Ce faisant elle maintient une liste U contenant les sommets qui appartiennent à l'enveloppe supérieure des points déjà visités. En pratique cette liste est implémentée par une pile, dans laquelle les sommets sont ordonnés de la droite vers la gauche. Ainsi, au début de la i-ème itération U contient l'enveloppe supérieure des points $p_1, \cdots, p_{i-1}$, et son premier élément u_1 n'est autre que p_{i-1}. La mise à jour de U lors de l'insertion du point p_i se fait en exploitant les observations simples suivantes (où U_r dénote l'enveloppe supérieure après la visite du point p_r) :

- le point p_i appartient forcément à U_i, étant le plus à droite au moment de son insertion ;
- un segment (u_k, u_{k+1}) de U_{i-1} appartient à U_i si et seulement si le triangle (p_i, u_k, u_{k+1}) est orienté dans le sens anti-horaire ;
- l'ensemble des segments de U_{i-1} qui doivent disparaître lors de l'insertion du sommet p_i forme une chaîne continue, dont le premier segment (s'il existe) est (u_1, u_2).

Théorème 1.19. *Le calcul de l'enveloppe convexe de n points dans le plan par l'algorithme de Graham nécessite un temps $O(n \log n)$.*

Démonstration. La validité de l'algorithme (le fait qu'il calcul l'enveloppe convexe correctement) peut se voir en imposant l'*invariant* suivant : à chaque étape du calcul de l'enveloppe convexe

2. Dans sa version originale, l'algorithme de Graham ne balayait pas les points horizontalement par une droite verticale, mais plutôt par une droite en rotation directe autour du point du nuage situé le plus bas – d'où son nom d'*algorithme de scan*. Andrew a proposé la variante par balayage horizontal, dans laquelle le tri initial des points se fait en comparant des abcisses plutôt que des angles, ce qui est plus sain du point de vue des erreurs d'arrondis comme on le verra dans la section 1.9.

Algorithm 2 Calcul de l'enveloppe supérieure par balayage (algorithme de Graham-Andrew)

```
 Entrée : S un ensemble de n points.
trier les points de S selon les coordonnées x croissantes;
initialiser U = [p_1, p_2]; // pile représentant l'enveloppe convexe
pour tout point p ∈ {p_3, ..., p_n} faire
    soient u_1 et u_2 les deux premiers sommets de U; // i.e. les plus à droite
    tant que (p, u_1, u_2) est orienté cw
        dépiler u_first de U
    empiler p dans U
retourner U
```

supérieure U tout triplet de sommets consécutifs (u_k, u_{k+1}, u_{k+2}) définit un triangle orienté en sens anti-horaire (un invariant similaire vaut pour l'enveloppe inférieure). Il suffit maintenant d'observer qu'après l'insertion d'un nouveau sommet p_i dans U_i, l'invariant ci-dessus reste satisfait, puisqu'on a supprimé tous les segments (u_k, u_{k+1}) qui définissent avec p_i un triangle orienté en sens horaire.

Pour évaluer le temps de calcul de l'enveloppe supérieure on va procéder de la manière suivante, en dénotant par D_i le nombre de segments $\{(u_1, u_2), \dots (u_{D_i}, u_{D_{i+1}})\}$ qui sont supprimés de U après l'insertion du sommet p_i. La i-étape de l'algorithme nécessite un temps (D_i+1), car il faut effectuer D_i tests d'orientations, chacun pouvant se faire en temps constant. La mise à jour de U se fait aussi en temps $O(D_i + 1)$, si on peut disposer d'une structure de donné permettant d'empiler et dépiler le premier élément en temps $O(1)$ et d'accéder au k-ième élément en temps $O(k)$ (une simple Pile fait l'affaire). Puisque chaque élément est inséré dans U exactement une fois, on a que $\sum_i^n (D_i + 1) = \Omega(n)$. Inversement, on observe que chaque sommet est supprimé au plus une fois à l'étape i, ne rentrant plus en jeu aux étapes successives. On peut donc amortir le coût des D_i suppressions et tests d'orientations sur toutes les n étapes : puisque au total on aura supprimé au plus $O(n)$ sommets, on a la borne supérieure suivante : $\sum_i^n (D_i + 1) = O(n)$. Et pour terminer il suffit de rappeler que la phase de prétraitement (le tri de n nombres) demande un temps $O(n \log n)$, alors que la fusion des deux enveloppes supérieure et inférieure peut se faire en temps $O(1)$. □

1.6.3 Approche par *Division-Fusion*

En s'inspirant de l'algorithme *MergeSort* il est possible de calculer l'enveloppe convexe par un procédé de Division-Fusion.

La première phase est le prétraitement des points $\{p_1, \dots, p_n\}$, que l'on trie selon les abscisses croissantes. Ensuite, à chaque étape les points sont partitionnés en deux sous-ensembles A et B de même taille environ séparables par une droite verticale. Pour ce faire on calcule le médian du nuage de points le long des abcsisses, puis on sépare les points selon que leur abcisse est supérieure (ensemble A) ou strictement inférieure (ensemble B) à celle du médian. Le calcul du médian se fait en temps linéaire par un simple parcours de la liste triée des points. Après l'appel récursif on dispose des enveloppes convexes $\mathcal{H}_A = conv(A)$ et $\mathcal{H}_B = conv(B)$. Il ne reste alors plus qu'à calculer les tangentes supérieure et inférieure qui serviront à trouver $\text{conv}(A \cup B)$.

Théorème 1.20. *L'algorithme par Division-Fusion calcule l'enveloppe convexe de n points en temps $O(n \log n)$.*

Démonstration. Le prétraitement des points prend temps $O(n \log n)$. Le partitionnement en sous-ensembles A et B nécessite temps $O(n)$, ainsi que le calcul des deux tangentes communes à

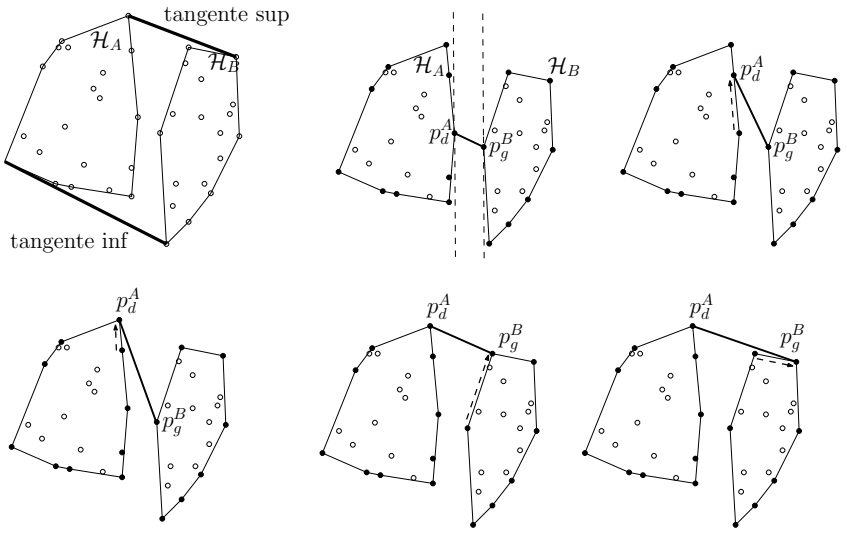


FIGURE 1.15 – *Calcul de l'enveloppe convexe par* Division-Fusion

Algorithm 3 Calcul de l'enveloppe convexe (Division-fusion)

```
Entrée : S ensemble trié de points.
si |S| ≤ 3 calculer et retourner l'enveloppe convexe de S;
sinon
    partitionner S en deux sous-ensembles (de même taille)
    Calculer H_A = conv(A) (appel récursif)
    Calculer H_B = conv(B) (appel récursif)
    Fusionner H_A et H_B
```

$\mathcal{H}_A$ et $\mathcal{H}_B$ (voir lemme ci-dessous). Aussi la mise à jour de l'enveloppe convexe, à partir de celles de A et B peut se faire en temps $O(n)$: cela consiste à fusionner deux listes chaînées et ajouter les deux tangentes aux résultat. Pour terminer, il reste à prendre en compte la complexité des appels récursifs. Si on dénote par $T(n)$ le temps de calcul pour un ensemble de n points, alors on peut écrire la récursion suivante

$$T(n) = \begin{cases} 1 \text{ si } n \leq 3 \\ O(n) + 2T(\frac{n}{2}) \text{ sinon} \end{cases}$$

qui coïncide avec la récursion de *MergeSort* et admet comme solution $T(n) = O(n \log n)$. □

Tangente supérieure. Il reste maintenant à décrire la procédure permettant de trouver la tangente supérieure de deux enveloppes convexes disjointes $\mathcal{H}_A$ et $\mathcal{H}_B$ (le pseudo-code est donné par l'algorithme 4). L'idée consiste à déterminer d'abord le point p_d^A plus à droite de l'ensemble A et le point plus à gauche p_g^B de l'ensemble B. On teste si le segment $\overline{p_d^A p_g^B}$ est la tangente supérieure : si ce n'est pas le cas on déplace l'extrémité gauche le long de $\mathcal{H}_A$ et l'extrémité de droite le long de $\mathcal{H}_B$. La procédure termine lorsque les deux extrémités définissent la tangente supérieure cherchée.

Lemme 1.21. *Étant donnés deux ensemble de points disjoints A et B (sépares par une droite verticale) et leurs enveloppes convexes respectives $\mathcal{H}_A$ et $\mathcal{H}_B$, le calcul de la tangente supérieure peut se faire en temps $O(|A| + |B|)$.*

Algorithm 4 Calcul de la tangente supérieure

```
 Entrée : H_A et H_B.
soit a le point le plus à droite de H_A;
soit b le point le plus à gauche de H_B;
Tant que ab n'est pas la tangente supérieure
    Tant que ab n'est pas une tangente supérieure à H_A
         a = a − 1; (déplacer a en sens horaire)
    Tant que ab n'est pas une tangente supérieure à H_B
         b = b + 1; (déplacer b en sens anti-horaire)
Retourner le segment ab.
```

Démonstration. On utilise un argument déjà mentionné au cours de la preuve du théorème 1.19, qui permet d'évaluer de manière amortie le coût de la marche le long des points extremaux de $\mathcal{H}_A$ et $\mathcal{H}_B$. En effet, il suffit de remarquer que chaque sommet, de $\mathcal{H}_A$ et $\mathcal{H}_B$, est visité au plus une seule fois. Ainsi, la boucle de l'algorithme 4 nécessite au plus $O(\mathcal{H}_B + \mathcal{H}_B) \leq |A| + |B|$ étapes : à chaque étape il reste juste à tester si le segment calculé est la tangente supérieure, ce qui revient à faire deux tests d'orientation, faisant intervenir au total quatre sommets. □

1.6.4 Borne inférieure sur la complexité

Théorème 1.22. *Étant donnés n points $\{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, le calcul de l'enveloppe convexe nécessite un temps $\Omega(n \log n)$.*

Démonstration. L'idée principale repose sur une réduction au problème du tri de n nombres réels $\{x_1, \ldots, x_n\}$,dont la complexité est connue et $\Omega(n \log n)$. Il suffit de considérer les points (x_i, x_i^2), qui appartiennent à la parabole $y = x^2$. Il est facile de voir que ces points sont en position convexe et appartiennent tous à l'enveloppe convexe inférieure, qui les relie dans l'ordre de leurs abcisses. Dès lors, en calculant la liste des sommets le long de l'enveloppe convexe inférieure on obtient la liste triée des nombres x_i. □

1.7 Enveloppes convexes en dimension supérieure

Les algorithmes de Jarvis et de division-fusion présentés dans la section 1.6 se généralisent en toutes dimensions (voir la figure 1.16 pour un exemple d'exécution de l'approche par division-fusion en 3D), avec des complexités plus ou moins bonnes suivant la dimension. Toutefois, aucun n'est optimal en toutes dimensions. Il existe bien d'autres approches pour le calcul de l'enveloppe convexe, dont certaines sont optimales en toutes dimensions. Mais au fait, qu'entend-on par *algorithme optimal*?

Théorème 1.23. *Le calcul de l'enveloppe convexe de n points en dimension d nécessite un temps $\Omega(n \log n + n^{\lfloor \frac{d}{2} \rfloor})$ dans le cas le pire.*

Le terme $\Omega(n \log n)$ dans la borne ci-dessus provient du fait qu'on peut toujours considérer des points dans le plan comme étant plongés dans $\mathbb{R}^d$: ainsi, le calcul de l'enveloppe convexe pour $d \geq 2$ est au moins aussi difficile qu'en dimension $d = 2$. Pour le terme restant, $\Omega(n^{\lfloor \frac{d}{2} \rfloor})$, il suffit de se rappeler le théorème de la borne supérieure (théorème 1.17), qui dit que l'enveloppe convexe de n points en dimension d a $\Omega(n^{\lfloor \frac{d}{2} \rfloor})$ faces dans le cas le pire.

Ainsi, par *algorithme optimal* nous entendons un algorithme dont la complexité coïncide avec la borne inférieure donnée ci-dessus. Le premier algorithme de ce type a été mis au point par Clarkson et Shor [CS89]. Il est randomisé et c'est sa complexité moyenne qui atteint la

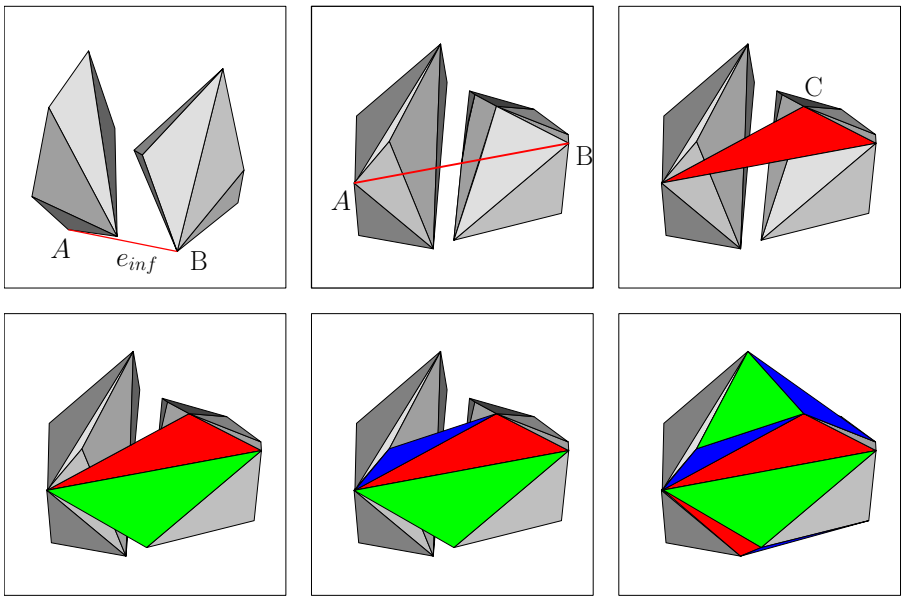


Figure 1.16 – *Calcul de l'enveloppe convexe 3D par l'algorithme de* Division-Fusion. *Une fois calculées récursivement les deux enveloppes convexes gauche et droite, il reste à calculer un cylindre défini par les tangentes communes. On commence par trouver la tangente inférieure commune e_{inf}, qui peut se calculer de manière similaire au cas planaire. Pour trouver un triangle tangent aux deux enveloppes, on peut observer qu'un tel triangle est défini par $e_{inf} = (A, B)$ et par un autre point C, appartenant soit à l'enveloppe de gauche soit à celle de droite : il suffit de tester les sommets adjacents à A ou B, ce qui peut se faire en temps $O(deg(A) + deg(B))$. Une fois déterminé le triangle tangent (A, B, C) on peut recommencer cette procédure, en partant de l'arête (A, C) (ou de l'arête (BC)) pour déterminer le prochain triangle tangent commun. Il est possible de montrer que le calcul de tous les triangles tangents communs prend temps $O(n)$. En observant que la récursion donnée par l'algorithme de division-fusion 3D est la même que dans le cas planaire, on peut arriver à une borne de $O(n \log n)$, qui est optimale.*

borne optimale. Notez que la randomisation intervient au niveau de l'ordre dans lequel les points de données sont traités, et que la complexité moyenne est calculée sur l'ensemble des permutations aléatoires de ces points. De fait l'analyse ne porte pas sur la distribution des points de données dans l'espace et ne fait aucune hypothèse particulière sur cette distribution. Le premier algorithme déterministe de complexité optimale a été proposé par Chazelle [Cha91] quelques années plus tard.

Théorème 1.24. *Le calcul de l'enveloppe convexe de n points en dimension d peut se faire de manière déterministe en temps optimal $O(n \log n + n^{\lfloor \frac{d}{2} \rfloor})$ dans le cas le pire.*

1.8 Arrangements de droites dans le plan

Définition 1.25. *Étant donné un ensemble de droites $L = \{l_1, \ldots, l_n\}$ dans le plan, on dénote par $\mathcal{A}(L)$ l'*arrangement *défini comme la subdivision du plan induite par les droites de L (l'ensemble de ses sommets, arêtes et faces).*

La *complexité* d'un arrangement est décrite par le nombre de ses sommets, arêtes et faces, et peut être évaluée à l'aide du théorème suivant :

Théorème 1.26. *Étant donné un arrangement $\mathcal{A}(L)$ de n droites $L = \{l_1, \ldots, l_n\}$, on a :*
- *le nombre des sommets de $\mathcal{A}(L)$ est au plus $\frac{n(n-1)}{2}$;*
- *le nombre des arêtes est borné par n^2 ;*
- *le nombre de cellules et au plus $\frac{n^2}{2} + \frac{n}{2} + 1$.*

Remarquons que même si la formule d'Euler ne peut pas s'appliquer directement (car un arrangement ne définit pas une carte planaire), le théorème précédent nous confirme l'existence d'une dépendance linéaire entre le nombre de sommets, arêtes et faces d'un arrangement.

Théorème de la zone. Étant donné un arrangement $\mathcal{A}(L)$, la *zone* d'une droite $l \in L$ dans $\mathcal{A}(L)$ est définie par le 2-complexe cellulaire constitué par les faces de $\mathcal{A}(L)$ qui intersectent l (ce 2-complexe incluant les sommets et arêtes incidentes à ses faces). Notons que même si la complexité d'un arrangement peut être quadratique, la complexité d'une zone (i.e. le nombre total de ses faces, arêtes et sommets) reste linéaire :

Théorème 1.27. *La taille de la zone d'une droite dans un arrangement de taille n est au plus $O(n)$.*

De ce résultat découle un algorithme incrémental de construction d'arrangements de droites dans le plan. Les droites sont insérées une par une et l'arrangement est mis à jour après chaque insertion. À l'étape i la zone de la droite à insérer a une taille linéaire $O(i-1)$ et peut être calculée avec la même complexité. Ainsi la complexité totale de l'algorithme est quadratique en le nombre de droites, ce qui est optimal dans le cas le pire puisque la taille de l'arrangement est quadratique.

Le théorème de la zone se généralise en toutes dimensions d, où il dit que la taille de la zone d'un hyperplan dans un arrangement de n hyperplans est $O(n^{d-1})$ [ESS93].

1.9 Robustesse des algorithmes géométriques

La correction des algorithmes géométriques repose sur des résultats théoriques énoncés et prouvés dans un système de coordonnées réelles. Or, tout ce que peut fournir l'arithmétique flottante des ordinateurs est un système de coordonnées discrètes, dans lequel les calculs ne peuvent être effectués de manière exacte. Les résultat de ces derniers sont donc approchés avec une certaine marge d'erreur, certes très petite, mais dont l'existence suffit à faire s'écrouler toutes nos constructions théoriques et à faire buguer nos programmes en pratique. Pour résoudre ce problème il est nécessaire de changer l'arithmétique utilisée pour les calculs. C'est ce que font toutes les bibliothèques de calcul géométrique modernes.

Pour illustrer cette idée, prenons l'exemple de l'algorithme de Jarvis présenté à la section 1.6.1. Si vous regardez le pseudo-code 1, vous distinguerez deux types de tâches effectuées par l'algorithme :
- Des tâches purement combinatoires, dont l'objectif est de construire une ou plusieurs structures de données et de les maintenir au fil du temps. Dans ce cas précis la structure maintenue est une liste doublement chaînée représentant les points sur le bord de l'enveloppe convexe.
- Des tâches plus algébriques, dont l'objectif est de répondre à certaines questions concernant les données. Dans ce cas précis ce sont les deux tests `si angle(.,.,.)<min`, qui répondent `true` si l'angle calculé est plus petit que la valeur stockée dans la variable `min`. Ces tests sont appelés des *prédicats* dans le jargon informatique.

Plus généralement, tout algorithme peut être divisé entre une partie combinatoire et une partie algébrique faite de prédicats et de constructions sur les données. Une particularité forte des algorithmes géométriques est de faire appel à des prédicats de nature géométrique (comparaison

d'angles ou de distances, tests d'orientation, tests d'appartenance à une boule circonscrite, etc.) qui sont potentiellement très complexes et dont la résolution exacte en arithmétique flottante est impossible. C'est le cas du test sur les angles dans l'algorithme de Jarvis, qui ne pourra être résolu que de manière approchée en pratique, ce qui induit des cas où l'angle sera inférieur à `min` mais sera détecté comme étant supérieur, et réciproquement. Pire : ici `min` est le résultat d'un précédent calcul d'angle, qui lui aussi etait approché. Ainsi, on en revient à comparer les résultats de deux calculs approximatifs. Imaginez les erreurs d'exécution que cela peut entraîner... Une illustration est donnée dans la figure 1.17. Notez qu'on n'a pas ces problèmes avec des algorithmes plus classiques comme les algorithmes de tri, où les seuls prédicats impliqués sont de simples comparaisons de nombres fournis en entrée.

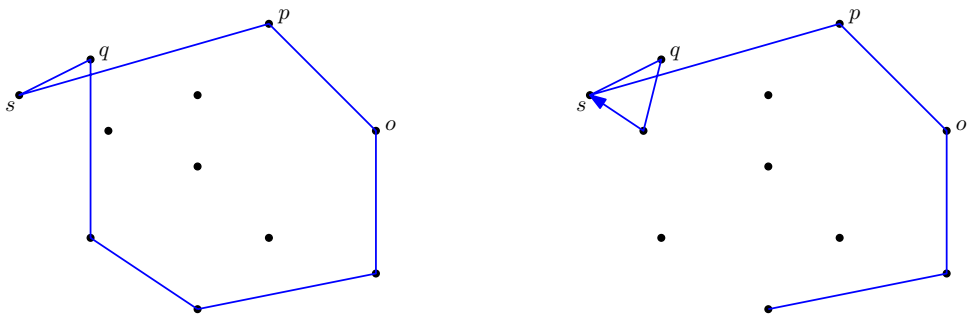


Figure 1.17 – *Deux exécutions erronées de l'algorithme de Jarvis. Dans les deux cas un seul test de comparaison d'angles est faux : celui entre les angles (op, pq) et (op, ps), qui sont très proches. Cette erreur fait choisir s et non q comme prochain sommet après p par l'algorithme. À l'itération suivante le point q devient celui qui minimise l'angle (il devient alors négatif, ce qui pourrait même faire crasher certaines implémentations) et devient donc le prochain sommet. Dans le premier cas de figure, le sommet suivant q se trouve être par bonheur sur l'enveloppe convexe, et le reste de l'exécution se déroule sans problème, donnant seulement un résultat faux. Dans le deuxième cas de figure le sommet suivant q n'est toujours pas sur le bord de l'enveloppe convexe, et in fine on retourne au point s ce qui fait entrer l'algorithme dans une boucle infinie.*

Que faire pour éviter que de tels cas pathologiques ne se produisent ? Tout d'abord, en ne calculant que des fonctions polynômiales à coefficients entiers pour effectuer les tests. Par exemple, dans l'algorithme de Jarvis, au lieu de comparer des angles obtenus par des arcsinus de produits vectoriels, on comparera directement les produits vectoriels en question. Ceci permet de représenter le test sous forme d'une inéquation polynômiale à coefficients entiers. Le résultat du test dépend du signe du polynôme mis en jeu dans l'équation, qui peut être évalué avec certitude de diverses façons, dont les suivantes :

- **Arithmétique exacte.** L'idée est d'utiliser une arithmétique multi-précision pour représenter les nombres exactement en mémoire et certifier les résultat des calculs. Ceci permet d'évaluer le polynôme du test exactement et donc d'obtenir son signe avec certitude. En Java on peut le faire grâce aux classes `BigInteger` et `BigDecimal`. Cette approche a l'avantage d'être facile à mettre en œuvre : il suffit grosso modo de remplacer les `int` par des `BigInteger` et les `double` par des `BigDecimal`. Le gros inconvénient est que, au fur et à mesure que l'algorithme se déroule et que les prédicats et les constructions s'empilent, la taille des représentations de nombres en mémoire augmente considérablement, ralentissant d'autant la vitesse des calculs. En pratique, le recours à une arithmétique exacte peut facilement ralentir l'exécution d'un algorithme d'un facteur 100 ou 1000. Cette approche n'est donc pas envisageable dans un produit logiciel fini, mais peut être utilisée temporairement pour tester expérimentalement la fiabilité d'un algorithme.

- **Filtrage statique.** Lorsqu'on a peu de tests différents à effectuer dans un algorithme (comme c'est le cas par exemple dans l'algorithme de Jarvis), on peut prendre le temps de calculer (à la main) une borne $\varepsilon > 0$ sur l'erreur commise lors du calcul en arithmétique flottante pour chaque test. Une fois la borne évaluée, on peut *filtrer* le test à l'exécution de l'algorithme comme suit : on évalue tout d'abord le polynôme de manière approchée en arithmétique flottante. Si le résultat tombe en-dehors de l'intervalle $[-\varepsilon, \varepsilon]$, alors on est sûr du signe du résultat et on peut donc le retourner. Sinon on a recours à la première approche, c'est-à-dire qu'on réévalue le polynôme de manière exacte pour déterminer son signe avec certitude. Cette approche est la plus efficace en pratique, à condition que les données soient suffisamment *génériques*, i.e. qu'il n'y ait pas trop de cas presque dégénérés où les polynômes sont quasiment nuls [3]. L'inconvénient majeur est qu'il faut calculer à la main une borne d'erreur pour chaque test qui soit suffisamment petite pour que le recours au calcul exact ne soit pas trop fréquent. Or, ceci peut prendre beaucoup de temps lorsque les prédicats sont complexes.
- **Arithmétique d'intervalle et filtrage dynamique.** Une version plus paresseuse de la deuxième approche consiste à évaluer la borne d'erreur à la volée pendant les calculs, en utilisant une représentation des nombres par des intervalles. Typiquement, un nombre x fourni en entrée est représenté par le singleton $\{x\}$ puisqu'il est connu exactement. Ensuite la largeur des intervalles augmente au fur et à mesure des calculs. Lors d'un test, on évalue le polynôme puis on regarde si l'intervalle de résultat contient zéro on non. Dans la négative, on peut retourner le signe du polynôme directement. Sinon on doit recourir à l'arithmétique exacte pour réévaluer le polynôme avec précision. Cette approche a une limitation évidente : la longueur des intervalles augmentant avec le temps, il se peut qu'elle devienne tellement grande au bout d'un moment que le recours à l'arithmétique exacte deviendrait systématique. Malgré tout, les temps de calcul observés en pratique sont généralement très bon, ce qui fait de cette approche un bon compromis entre temps de développement et efficacité réelle. C'est pourquoi elle est utilisée dans plusieurs bibliothèques de calcul géométrique, comme la bibliothèque CGAL [4].

Comme on le voit, toutes ces approches présentent des avantages et des inconvénients différents. Le choix d'une approche particulière en pratique dépend donc du contexte dans lequel l'algorithme doit être utilisé.

3. Sinon on passe son temps à évaluer les polynômes deux fois : l'une de manière approchée, l'autre de manière exacte, ce qui prend encore plus de temps que la première approche.

4. `http://www.cgal.org`

Bibliographie

[BD98] Prosenjit Bose and Luc Devroye. Intersections with random geometric objects. *Comput. Geom.*, 10(3) :139–154, 1998.

[BDP+02] J.-D. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec. Triangulations in cgal. *Comput. Geom. Theory Appl.*, 22 :5–19, 2002.

[Cha91] B. Chazelle. An optimal convex hull algorithm and new results on cuttings. *Foundations of Computer Science, Annual IEEE Symposium on*, 0 :29–38, 1991.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, ii. *Discrete Comput. Geom.*, 4(5) :387–421, 1989.

[ESS93] Herbert Edelsbrunner, Raimund Seidel, and Micha Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22(2) :418–429, 1993.

[Gra72] Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4) :132–133, 1972.

[GS85] Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and computation of voronoi diagrams. *ACM Trans. Graph.*, 4(2) :74–123, 1985.

[Jar73] R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Inf. Process. Lett.*, 2(1) :18–21, 1973.

[Ket99] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Comput. Geom.*, 13(1) :65–90, 1999.

[LW69] A. T. Lundell and S. Weingram. *The Topology of CW Complexes*. Van Nostrand Reinhold Company, New York, 1969.

[McM70] P. McMullen. The maximum number of faces of a convex polytope. *Mathematika*, 17 :179–184, 1970.

Chapitre 2

Triangulations de Delaunay et diagrammes de Voronoï

Dans le chapitre 1 nous avons défini les triangulations du plan simplement comme des ensembles maximaux d'arêtes d'intérieurs deux-à-deux disjoints. Cette définition ne se s'étend pas directement en dimensions supérieures : en effet, en dimensions 3 et plus, génériquement tous les segments reliant des paires de points quelconques du nuage sont d'intérieurs deux-à-deux disjoints. Or, ils ne définissent clairement pas tous ensemble le squelette d'une triangulation.

Dans la section 2.1 nous allons donner une nouvelle version de la définition 1.3 qui se généralise en toutes dimensions. Nous allons également parler d'une classe particulière de triangulations appelées triangulations de Delaunay, et nous établirons le lien entre ces triangulations et certaines décompositions de l'espace ambiant appelées diagrammes de Voronoï. Ensuite nous évoquerons certaines des propriétés remarquables des triangulations de Delaunay qui ont fait leur succès dans de nombreux domaines d'application (section 2.2).

2.1 Définitions

On rappelle (voir la section 1.1.5) qu'un k-simplexe plongé dans $\mathbb{R}^d$ ($k \leq d$) est l'enveloppe convexe de $k+1$ points de $\mathbb{R}^d$ qui sont affinement indépendants. Une l-face du simplexe ($l \leq k$) est l'enveloppe convexe d'un sous-ensemble de $l+1$ sommets du simplexe. Une l-face est dite *propre* dès que $l < k$. Le bord du simplexe est l'union de ses faces propres, tandis que l'intérieur relatif est le complémentaire de cette union.

Un complexe simplicial plongé dans $\mathbb{R}^d$ est un ensemble de simplexes qui est clos par inclusion (i.e. les faces de chaque simplexe sont des simplexes du complexe) et par intersection (i.e. l'intersection de deux simplexes distincts du complexe est une face propre commune). En outre, ces conditions impliquent que les intérieurs relatifs de deux simplexes distincts quelconques du complexe sont disjoints.

La notion de complexe simplicial plongé nous permet de définir les triangulations en toutes dimensions :

Définition 2.1. *Étant donné un nuage (fini) de points P dans $\mathbb{R}^d$, une* triangulation *de P est un complexe simplicial plongé dont les sommets sont dans P et dont l'union des simplexes coïncide avec l'enveloppe convexe de P.*

Remarquez qu'on ne demande pas que tous les points de P soient sommets du complexe (seuls les points du bord de l'enveloppe convexe doivent impérativement l'être) : il se peut en effet que certains points de P soient ignorés, comme c'est le cas par exemple avec les *triangulations*

régulières, une généralisation des triangulations de Delaunay où les sommets ont des poids et que vous verrez en cours de Master 2.

2.1.1 Triangulations de Delaunay

Commençons par définir ce qu'est la propriété de Delaunay pour un simplexe :

Définition 2.2. *Étant donné un nuage de points P dans $\mathbb{R}^d$, un simplexe dont les sommets sont dans P est dit* de Delaunay *s'il existe une boule circonscrite au simplexe dont l'intérieur est vide de points de P. Une telle boule est appelée* boule de Delaunay.

L'ensemble des simplexes de Delaunay est appelé *triangulation de Delaunay*. Ce terme est en fait abusif car cet ensemble ne définit pas toujours une triangulation : en effet, imaginons que tous les points de P se situent sur un même cercle dans le plan. Alors toutes les arêtes joignant deux points quelconques de P sont de Delaunay et donc le squelette de la triangulation est le graphe complet, qui ne peut être plongé dans le plan dès que P est constitué de 4 points en position convexe. Toutefois, cet exemple n'est pas générique dans le sens qu'on peut perturber les points de P infinitésimalement pour briser la co-circularité et ainsi réduire considérablement le nombre d'arêtes. Nous dirons que le nuage de points P est *en position générale* dans $\mathbb{R}^d$ quand aucun sous-ensemble de $d+2$ points de P n'est co-sphérique. Quand P n'est pas en position générale, il est facile de l'y placer par une perturbation infinitésimale de ses points.

Théorème 2.3. *Quand un nuage de points P est en position générale, l'ensemble des simplexes de Delaunay forme une triangulation de P, appelée* triangulation de Delaunay.

La preuve de ce résultat s'appuie sur un outil théorique fondamental : le relèvement des points dans l'*espace des sphères* $\mathbb{R}^{d+1}$. Étant donné un point $x \in \mathbb{R}^d$, nous relevons x dans $\mathbb{R}^{d+1}$ par la fonction $x \mapsto (x, x^2)$ qui paramétrise le paraboloïde unité.

Lemme 2.4. *Dans l'espace des sphères $\mathbb{R}^{d+1}$, l'intersection d'un hyperplan H non vertical avec le paraboloïde unité est soit vide, soit le relevé d'une sphère de $\mathbb{R}^d$. L'intersection du demi-espace inférieur H^- avec le paraboloïde est alors le relevé de la boule bordée par cette sphère.*

Démonstration. Soit $\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_d x_d + \alpha_{d+1} x_{d+1} = \beta$ l'équation de l'hyperplan H. Comme l'hyperplan n'est pas vertical, on a $\alpha_{d+1} \neq 0$ et, quitte à multiplier les deux membres de l'équation par un même facteur, on peut supposer sans perte de généralité que $\alpha_{d+1} = 1$. Un point (x, x^2) du paraboloïde se trouve alors dans le demi-espace inférieur H^- si et seulement si

$$\sum_{i=1}^{d} \alpha_i x_i + x^2 \leq \beta \Leftrightarrow x^2 - 2x \cdot (-\frac{\alpha_1}{2}, \cdots, -\frac{\alpha_d}{2}) + (-\frac{\alpha_1}{2}, \cdots, -\frac{\alpha_d}{2})^2 \leq \beta + (-\frac{\alpha_1}{2}, \cdots, -\frac{\alpha_d}{2})^2, \quad (2.1)$$

ce qui signifie que x se trouve dans la boule B de centre c et rayon r dans $\mathbb{R}^d$, où $c = -(\frac{\alpha_1}{2}, \cdots, \frac{\alpha_d}{2}) \in \mathbb{R}^d$ et $r = \sqrt{\beta + c^2}$. Lorsque (x, x^2) se trouve sur l'hyperplan H l'inégalité de l'équation 2.1 se transforme en égalité et ainsi x se situe sur la sphère ∂B. Notez que la boule B est non vide précisément quand $\beta + c^2 \geq 0$, ce qui correspond à la condition sous laquelle l'hyperplan H (et donc aussi le demi-espace inférieur H^-) intersect le paraboloïde. □

Avec ce lemme technique à disposition, nous pouvons maintenant démontrer le théorème 2.3 :

Preuve du théorème 2.3. Soient $p_0, \cdots, p_k$ des points du nuage P ($k \leq d$). Leurs relevés sur le paraboloïde forment une face de l'enveloppe convexe inférieure du relevé de P si et seulement s'il existe un hyperplan H dans $\mathbb{R}^{d+1}$ qui contient $(p_0, p_0^2), \cdots, (p_k, p_k^2)$ et tel que l'intérieur du demi-espace inférieur H^- est vide de points du relevé de P. D'après le lemme 2.4, ceci

est équivalent à dire qu'il existe dans $\mathbb{R}^d$ une boule circonscrite à $p_0, \cdots, p_k$ dont l'intérieur ne contient aucun point de P. Ainsi, les faces de l'enveloppe convexe inférieure du relevé de P sont en bijection avec l'ensemble des simplexes de Delaunay de P. Lorsque ce dernier est en position générale, il ne contient aucun sous-ensemble de $d+2$ points co-sphériques, i.e. son relevé ne contient aucun sous-ensemble de $d+2$ points situés à l'intersection du paraboloïde avec un même hyperplan. Dès lors, l'enveloppe convexe inférieure du relevé de P est définie de manière unique et est simpliciale. De plus, elle se projette verticalement de manière bijective sur l'enveloppe convexe de P dans $\mathbb{R}^d$. Ainsi, son image par la projection, i.e. l'ensemble des simplexes de Delaunay, est une triangulation de P. □

Notez que la triangulation de Delaunay a tous les points de P pour sommets. En effet, tout point de P est centre d'une boule de rayon nul qui le circonscrit et forme donc un simplexe de Delaunay. De manière équivalente, comme le paraboloïde dans l'espace des sphères $\mathbb{R}^{d+1}$ est convexe, tous les points du relevé de P se trouvent sur l'enveloppe convexe inférieure et donc en sont sommets.

2.1.2 Diagrammes de Voronoï

Les triangulations de Delaunay ont un lien fondamental avec certaines décompositions de l'espace ambiant appelées diagrammes de Vororoï.

Définition 2.5. *Étant donné un nuage de points P dans $\mathbb{R}^d$, le* diagramme de Voronoï $\mathrm{Vor}(P)$ *est une décomposition cellulaire de l'espace $\mathbb{R}^d$, où la cellule de chaque point $p \in P$ est définie comme le lieu des points de $\mathbb{R}^d$ au moins aussi proches de p que du reste de P.*

Notez que les cellules du diagramme $\mathrm{Vor}(P)$ sont fermées par définition. Elles ne peuvent donc partitionner l'espace $\mathbb{R}^d$, bien qu'elles le couvrent. Pour tout $k \leq d$, une k-face du diagramme $\mathrm{Vor}(P)$ est l'intersection de $d+1-k$ cellules. D'après la définition 2.5, c'est également le lieu des points de $\mathbb{R}^d$ qui sont centres de boules de Delaunay circonscrites au simplexe formé par les centres des $d+1-k$ cellules. Par exemple, les cellules sont les d-faces du diagramme, et chaque cellule est bien le lieu des points de $\mathbb{R}^d$ qui sont centres de boules circonscrites au centre de la cellule et dont l'intérieur ne contient aucun point de P.

Le terme *k-face* provient du fait que k est la dimension intrinsèque de la face, quand P est en position générale. En effet, pour $k = d$ la face est une cellule, dont la dimension est d puisque les points de P sont tous distincts. Pour $k = d-1$ la face est incluse dans l'hyperplan médiateur de 2 points de P, qui est de dimension intrinsèque $d-1$. Pour k plus petit, la face est l'intersection de $d+1-k$ cellules de Voronoï et donc génériquement de $d-k$ hyperplans affinement indépendants, ce qui fait qu'elle est incluse dans un sous-espace affine de dimension $d-(d-k)=k$. Ainsi, toute k-face de $\mathrm{Vor}(P)$ a dimension k et est naturellement associée à un $(d-k)$-simplexe de $\mathrm{Del}(P)$. Cette association entre l'ensemble des faces de $\mathrm{Vor}(P)$ et l'ensemble des simplexes de $\mathrm{Del}(P)$ est bijective. En cela, les deux objets sont considérés comme duaux l'un de l'autre. Voir la figure 2.1 pour une illustration.

2.2 Quelques propriétés remarquables

2.2.1 Localement Delaunay contre globalement Delaunay

Ici nous nous concentrons sur les triangulations de Delaunay dans le plan.

Définition 2.6. *Étant donné une triangulation dans le plan, une paire de triangles (a,b,c) et (b,c,d) dans la triangulation est dite* localement de Delaunay *si le disque ouvert circonscrit à*

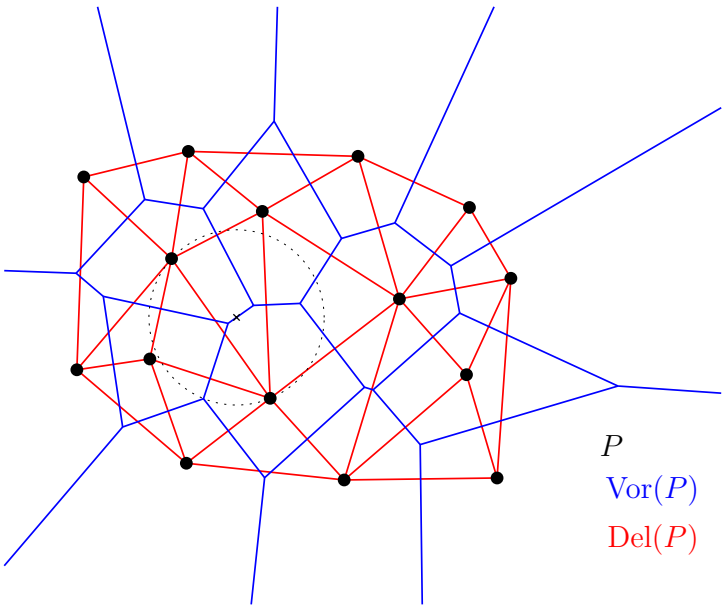


FIGURE 2.1 – *Diagramme de Voronoï d'un nuage de points P dans le plan, et sa triangulation de Delaunay duale. La dualité est indiquée par le disque de Delaunay marqué en pointillé et centré sur la croix.*

(a, b, c) ne contient pas d (ou, de manière équivalente, si le disque ouvert circonscrit à (b, c, d) ne contient pas a). La triangulation elle-même est dite localement de Delaunay *si toutes les paires de triangles partageant une arête sont localement de Delaunay.*

La définition 2.6 fournit un critère local pour déterminer si une triangulation est de Delaunay. En effet,

Théorème 2.7. *Si une triangulation dans le plan est localement de Delaunay, alors elle est (globalement) de Delaunay.*

Démonstration. Soit P un nuage de points dans le plan, et soit $\mathcal{T}$ une triangulation de P qui est localement de Delaunay. Supposons par contradiction qu'elle ne soit pas gobalement de Delaunay. Alors il existe un triangle (a, b, c) et un sommet v de $\mathcal{T}$ tels que v se trouve dans l'intérieur du disque circonscrit au triangle. Notez que v ne peut se trouver à l'intérieur du triangle puisqu'autrement la triangulation $\mathcal{T}$ ne pourrait avoir (a, b, c) pour simplexe. Considérons alors une arête de (a, b, c) dont la droite support sépare v de (a, b, c). Cette arête est commune à (a, b, c) et à un autre triangle de $\mathcal{T}$, mettons (b, c, d), qui se situe de fait du même côté de la droite support de l'arête que v. Comme la paire (a, b, c) et (b, c, d) est localement de Delaunay, le disque circonscrit à (b, c, d) contient le disque circonscrit à (a, b, c) de ce côté-ci de la droite support de l'arête, et donc son intérieur contient également v. Voir la figure 2.2 (gauche) pour une illustration. Ainsi on a identifié un triangle de $\mathcal{T}$ qui est différent de (a, b, c) et dont le disque circonscrit contient aussi v dans son intérieur. Notez que v ne peut être sommet de ce nouveau triangle car ses sommets se trouvent sur le bord du disque. En répétant ce processus un nombre arbitraire de fois, on construit une séquence arbitrairement longue de triangles de $\mathcal{T}$ qui sont **distincts** les uns des autres, chacun étant contenu dans une intersection de demi-plans ne contenant aucun des triangles le précédant dans la séquence. Ceci soulève une contradiction dès que la longueur de la séquence dépasse le nombre de triangles de $\mathcal{T}$. □

Ainsi, le critère local donné par la définition 2.6 permet de vérifier aisément si une triangulation plane $\mathcal{T}$ est de Delaunay ou non : il suffit pour cela d'itérer sur les arêtes intérieures de

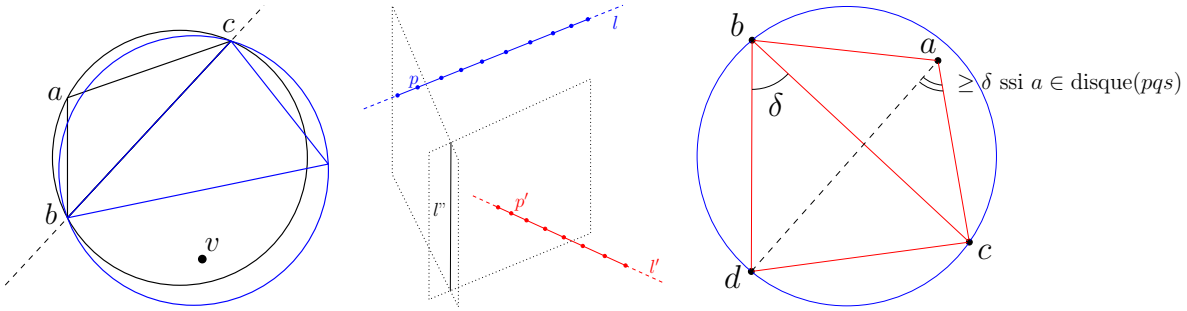


Figure 2.2 – *Gauche : pour la preuve du théorème 2.7. Milieu : cas où* Del(P) *a une taille quadratique dans* $\mathbb{R}^3$*. Droite : pour la preuve du théorème 2.9.*

la triangulation, et pour chacune d'entre elles de vérifier si le critère local est vérifié ou non. Le temps de calcul est linéaire en la taille de la triangulation.

La question est maintenant de savoir si ce critère permet de rendre une triangulation plane quelconque $\mathcal{T}$ de Delaunay par des bascules d'arêtes. L'algorithme est le suivant : tant qu'il reste des paires de triangles adjacents qui ne sont pas localement de Delaunay, on choisit arbitrairement l'une de ces paires (mettons (a, b, c) et (b, c, d)) et on *bascule* l'arête (b, c) en la remplaçant par l'autre diagonale (a, d).

Théorème 2.8. *Quel que soit l'ordre choisi pour le traitement des paires de triangles, l'algorithme ci-dessus termine et débouche sur une triangulation qui est globalement de Delaunay. Le nombre total de flips effectués est* $O(n^2)$*, où* n *est le nombre de sommets de la triangulation.*

La preuve de ce résultat sera admise ici — voir par exemple la section 9.3 de [dBvKOS00] pour les détails. Notez que la preuve ne marche qu'en 2 dimensions. La question de savoir si l'algorithme de bascule fonctionne en dimension 3 ou plus reste ouverte à ce jour.

2.2.2 Optimisation de l'angle minimal et de la séquence des angles

Ici encore on s'intéresse aux triangulations de Delaunay dans le plan. Étant donné une triangulation plane $\mathcal{T}$ quelconque (de Delaunay ou non), on désigne par *séquence des angles de* $\mathcal{T}$ la séquence des mesures des angles internes des triangles de $\mathcal{T}$, ordonnée de manière croissante. L'*angle minimal de* $\mathcal{T}$ est le plus petit élément dans cette séquence. Comme deux triangulations planes partageant le même ensemble de sommets ont toujours le même nombre de triangles (par la relation d'Euler, voir la section 1.2), on peut comparer leur séquences d'angles pour l'ordre lexicographique.

Théorème 2.9. *Étant donné un nuage de points* P *dans le plan,* Del(P) *est la triangulation d'ensemble de sommets* P *qui maximise le plus petit angle. C'est également celle qui maximise l'ordre lexicographique sur la séquence des angles.*

Démonstration. Soit $\mathcal{T}$ une triangulation ayant tous les points de P comme sommets. Supposons que cette triangulation ne soit pas de Delaunay. Alors par le théorème 2.7 elle n'est pas localement de Delaunay, i.e. il existe une paire de triangles adjacents (a, b, c) et (b, c, d) telle que le sommet a se situe à l'intérieur du disque circonscrit à (b, c, d), comme illustré dans la figure 2.2 (droite). L'angle interne $\widehat{cad}$ est alors plus grand que l'angle $\widehat{cbd}$ car tous deux s'appuient sur le même arc de cercle $\overline{cd}$. De même, l'angle $\widehat{bad}$ est plus grand que l'angle $\widehat{bcd}$. De manière similaire, on a $\widehat{bda} \geq \widehat{bca}$ et $\widehat{cda} \geq \widehat{cba}$. En conséquence, en basculant l'arête (b, c)

pour la remplacer par (a, d) on augmente le plus petit angle dans le quadrangle (a, b, c, d), sans modifier les autres triangles de la triangulation. En répétant cette opération comme dans la section 2.2.1, on transforme $\mathcal{T}$ en une triangulation de Delaunay par le théorème 2.8, sans diminuer le plus petit angle de la triangulation. Ceci étant vrai pour toute triangulation $\mathcal{T}$ de sommets les points de P, on conclut que parmi ces triangulations $\mathrm{Del}(P)$ est celle qui maximise le plus petit angle. Le même argument montre que $\mathrm{Del}(P)$ maximise en fait l'ordre lexicographique sur la séquence des angles de la triangulation. □

2.2.3 Arbre couvrant minimal

On rappelle qu'un arbre couvrant un nuage de points P dans $\mathbb{R}^d$ est un graphe connexe sans cycles qui a P pour ensemble de sommets. Si P possède n points, alors l'arbre possède exactement $n-1$ arêtes. Un arbre convrant minimal de P est un arbre couvrant dont la somme des longueurs des arêtes est minimale.

Théorème 2.10. *Étant donné un nuage de points P dans $\mathbb{R}^d$, toutes les arêtes des arbres couvrants minimaux de P sont des arêtes de* $\mathrm{Del}(P)$.

La preuve du théorème s'appuie sur le petit lemme technique suivant, qui a un intérêt en soi :

Lemme 2.11. *Soient P, Q deux ensembles finis de points disjoints dans $\mathbb{R}^d$. Toute paire de points $(p, q) \in P \times Q$ pour laquelle la distance $\|p-q\|$ est minimale forme une arête de* $\mathrm{Del}(P \cup Q)$.

Démonstration. Par hypothèse q est un plus proche voisin de p dans Q, donc la boule ouverte B_p de centre p et de rayon $\|p-q\|$ ne contient aucun point de Q. Symétriquement, la boule ouverte B_q de centre q et de rayon $\|p-q\|$ ne contient aucun point de P. Il s'ensuit que $B_p \cap B_q$ ne contient aucun point de $P \cup Q$. Or, la région $B_p \cap B_q$ contient la boule diamétrale du segment de droite $[p, q]$. Cette boule diamétrale est donc une boule de Delaunay, et de ce fait l'arête (p, q) est dans $\mathrm{Del}(P \cup Q)$. □

La preuve du théorème 2.10 découle aisément du lemme technique :

Preuve du théorème 2.10. Soit (p_1, p_2) une arête d'un arbre couvrant minimal A de P. Privé de cette arête, A devient une forêt avec deux arbres disjoints, A_1 et A_2. Soient P_1 et P_2 les ensembles de sommets de ces deux arbres : ils sont aussi disjoints, et l'un (disons P_1) contient p_1 tandis que l'autre contient p_2. La distance $\|p_1 - p_2\|$ réalise le minimum de la distance entre P_1 et P_2, car sinon il suffirait de remplacer l'arête (p_1, p_2) par une arête reliant une autre paire de points plus rapprochés dans $P_1 \times P_2$ pour construire un arbre couvrant de poids total strictement inférieur à celui de A. Il s'ensuit par le lemme 2.11 que l'arête (p_1, p_2) est dans $\mathrm{Del}(P_1 \cup P_2) = \mathrm{Del}(P)$. □

Ainsi, pour calculer un arbre couvrant minimal A d'un nuage de points P dans $\mathbb{R}^d$, on peut d'abord calculer (le squelette de) la triangulation de Delaunay de P puis rechercher les arêtes de A parmi celles de $\mathrm{Del}(P)$ en utilisant par exemple l'algorithme de Kruskal. Cette méthode a un intérêt algorithmique évident dans le plan, où la taille de $\mathrm{Del}(P)$ est linéaire par rapport à celle de P, par la relation d'Euler (voir le corollaire 1.7 dans la section 1.2). Comme on le verra plus loin, le calcul de $\mathrm{Del}(P)$ dans le plan peut se faire en temps $O(n \log n)$, où n est la taille de P. Ainsi, l'arbre couvrant minimal peut être calculé en temps optimal $O(n \log n)$, ce qui améliore significativement la borne $O(n^2)$ de l'algorithme de Kruskal.

Toutefois, ce résultat n'est pas valable en dimension 3 ou plus, où le nombre d'arêtes de $\mathrm{Del}(P)$ est $\Omega(n^2)$ dans le cas le pire. Pour le voir, il suffit d'échantillonner les points de P le long de deux droites l, l' non coplanaires dans $\mathbb{R}^3$, de manière à ce que la moitié des points se

trouvent sur l et l'autre moitié sur l', comme illustré sur la figure 2.2 (milieu). Dès lors, pour toute paire de points $(p, p') \in (P \cap l) \times (P \cap l')$, l'arête (p, p') est dans $\mathrm{Del}(P)$. En effet, les droites l, l' étant non parallèles, les plans normaux à l en p et à l' en p' s'intersectent le long d'une droite l''. Tout point sur cette droite a p comme plus proche voisin sur l et p' comme plus proche voisin sur l'. Maintenant, les droites l, l' étant non coplanaires, la droite joignant p à p' n'est pas située dans le plan vectoriel engendré par l, l', et donc l'hyperplan médiateur de p, p' ne peut être parallèle à l''. Dès lors, cette droite coupe l'hyperplan médiateur en un point c qui est équidistant de p et de p', avec p et p' comme plus proches voisins sur l et sur l' respectivement. La boule ouverte B_c de centre c et de rayon $\|c-p\| = \|c-p'\|$ ne contient donc aucun point de $P \subset l \cup l'$, tandis que son bord contient p et p'. B_c est donc une boule de Delaunay, et (p, p') une arête de $\mathrm{Del}(P)$. Comme ceci vaut pour toute paire $(p, p') \in (P \cap l) \times (P \cap l')$, on conclut que $\mathrm{Del}(P)$ possède $\Omega(n^2)$ arêtes.

2.3 Taille et calcul de la triangulation de Delaunay

2.3.1 Taille de la triangulation

Nous avons vu dans la section 2.2.3 que la triangulation de Delaunay peut avoir un nombre quadratique d'arêtes (et donc de simplexes) dans $\mathbb{R}^3$. Plus généralement,

Théorème 2.12. *La taille (en nombre de simplexes) de la triangulation de Delaunay de n points dans $\mathbb{R}^d$ est $\Theta(n^{\lceil \frac{d}{2} \rceil})$.*

Ce résultat découle du théorème de la borne supérieure (théorème 1.17). En effet, via le relèvement dans l'espace des sphères $\mathbb{R}^{d+1}$, la triangulation de Delaunay devient une enveloppe convexe inférieure, donc sa taille est au plus $O(n^{\lfloor \frac{d+1}{2} \rfloor}) = O(n^{\lceil \frac{d}{2} \rceil})$ par le théorème 1.17. Pour obtenir une borne inférieure on distingue le cas où la dimension d est paire de celui où elle est impaire. Quand d est pair, il suffit d'échantillonner les sommets de la triangulation sur la courbe des moments $t \mapsto (t, t^2, \cdots, t^d)$. D'après le théorème 1.17, la taille de l'enveloppe convexe (et donc aussi celle de la triangulation de Delaunay) devient $\Omega(n^{\lfloor \frac{d}{2} \rfloor}) = \Omega(n^{\lceil \frac{d}{2} \rceil})$. Reste à traiter le cas où d est impair, pour lequel il reste un facteur n à combler entre notre borne inférieure et notre borne supérieure. Ce cas se résout par un traitement similaire, avec toutefois quelques détails techniques supplémentaires que nous omettons ici (voir par exemple l'exercice 7.11 de [BY02]) : grosso modo, on échantillonne n points le long de la courbe paramétrée par $t \mapsto \frac{2}{d+1}(\cos(t), \sin(t), \cos(2t), \sin(2t), \cdots, \cos(\frac{d+1}{2}t), \sin(\frac{d+1}{2}t))$, qui est dessinée sur la sphère unité $\mathbb{S}^d \subset \mathbb{R}^{d+1}$. Par un argument similaire à celui utilisé pour la courbe des moments, on montre que l'enveloppe convexe des points échantillonnés a $\Omega(n^{\lfloor \frac{d+1}{2} \rfloor})$ faces au total. On envoie ensuite les échantillons sur le paraboloïde unité $x \mapsto (x, x^2)$ dans $\mathbb{R}^{d+1}$ par l'application projective qui envoie le centre de la sphère à l'infini le long de l'axe (O, x_d), ce qui préserve l'enveloppe convexe inférieure. On peut ensuite projeter les échantillons verticalement sur un nuage de points $P \subset \mathbb{R}^d$, dont la triangulation de Delaunay a $\Omega(n^{\lfloor \frac{d+1}{2} \rfloor}) = \Omega(n^{\lceil \frac{d}{2} \rceil})$ faces au total par construction.

Note culturelle 2.13. *La borne du théorème 2.12 est en fait valable pour toute triangulation de n points en position générale dans $\mathbb{R}^d$. Ce résultat, démontré par Rotschild et Straus [RS85], découle d'une variante du théorème de la borne supérieure qui dit que les complexes simpliciaux à n sommets homéomorphes à une d-sphère ont une taille $O(n^{\lceil \frac{d}{2} \rceil})$ [Sta75].*

2.3.2 Calcul de la triangulation

D'après le théorème 2.12 et le chapitre 1, on peut calculer la structure combinatoire de la triangulation de Delaunay de n points dans $\mathbb{R}^d$ en temps $O(n \log n + n^{\lceil \frac{d}{2} \rceil})$, simplement en relevant les points sur le paraboloïde unité dans l'espace des sphères $\mathbb{R}^{d+1}$ puis en calculant leur enveloppe convexe inférieure.

En classe nous verrons un algorithme plus direct, mis au point par Guibas et Stolfi [GS85] et basé sur une approche diviser-pour-régner. Dans ce poly nous présentons encore une autre méthode, appelée *méthode incrémentale*, qui permet non seulement de calculer la triangulation de Delaunay d'un nuage de points donné, mais également de la maintenir lorsque des points sont insérés ou supprimés du nuage, ce qui peut être très utile en pratique, par exemple dans le cadre de la génération de maillages ou de la reconstruction de formes (voir à ce propos le chapitre 4).

L'algorithme incrémental. Soit P un nuage de points dans $\mathbb{R}^d$ dont la triangulation de Delaunay a déjà été calculée. Soit $p \in \mathbb{R}^d$ un nouveau point à insérer dans P. Mettre à jour la triangulation revient à supprimer les simplexes de $\mathrm{Del}(P)$ qui ne sont plus de Delaunay à cause de l'insertion de p, puis à ajouter tous les nouveaux simplexes de Delaunay créés par l'insertion.

Pour simplifier l'exposition, on supposera que p est situé dans l'enveloppe convexe de P, le cas où p se situe à l'extérieur nécessitant quelques aménagements supplémentaires. De plus, on supposera que P contient au moins $d+1$ points affinement indépendants, de manière à ce que $\mathrm{Del}(P)$ soit d-dimensionnel. Les cas où $\mathrm{Del}(P)$ n'est pas d-dimensionnel se traitent de manière similaire, les points du nuage étant situés dans un sous-espace affine de $\mathbb{R}^d$.

On appelle *zone de conflit* de p, notée $C(p)$, l'ensemble des simplexes de $\mathrm{Del}(P)$ pour lesquels il existe au moins une boule de Delaunay circonscrite dont l'intérieur contient p. Ces simplexes sont dits *en conflit* avec p. La zone de conflit contient entre autres tous les simplexes de $\mathrm{Del}(P)$ qui doivent disparaître de la triangulation lors de l'insertion de p (voir la figure 2.3 pour une illustration).

Lemme 2.14. *$C(p)$ est un sous-complexe simplicial connexe de $\mathrm{Del}(P)$. De plus, l'union des simplexes $C(p)$ est un sous-ensemble de $\mathbb{R}^d$ qui est étoilé par rapport à p.*

Démonstration. Soit σ_p un simplexe de $\mathrm{Del}(P)$ contenant p. Considérons un simplexe σ en conflit avec p et un point x quelconque dans l'intérieur relatif de σ. Alors par le même raisonnement que dans la preuve du théorème 2.7, tous les simplexes de $\mathrm{Del}(P)$ intersectés par le segment de droite $[p, x]$ (entre autres σ et σ_p) sont en conflit avec p. Ainsi, le segment est inclus dans l'union des simplexes de $C(p)$, ce qui montre que cette union est étoilée par rapport à p. De plus, le simplexe σ peut être atteint à partir de σ_p par une marche dans la triangulation le long du segment $[p, x]$, qui d'après ce qu'on vient de voir ne traverse que des simplexes en conflit avec p. Ainsi, $C(p)$ est un complexe simplicial connexe (par arcs issus de σ_p). □

Dans la suite on désignera sous le terme d'*intérieur* de la zone de conflit de p l'ensemble des simplexes formant l'intérieur du complexe $C(p)$ muni de la topologie usuelle de $\mathbb{R}^d$. Autrement dit, un simplexe σ est dans l'intérieur de la zone de conflit si et seulement si σ n'est pas sur le bord de l'enveloppe convexe de P et tous les d-simplexes de $\mathrm{Del}(P)$ incidents à σ sont dans $C(p)$. En particulier, tout d-simplexe de $C(p)$ est considéré comme étant dans l'intérieur de $C(p)$.

Lemme 2.15. *Les simplexes de $\mathrm{Del}(P)$ qui disparaissent de la triangulation lors de l'insertion de p sont précisément ceux appartenant à l'intérieur de la zone de conflit de p.*

Démonstration. Soit σ un simplexe de $\mathrm{Del}(P)$ appartenant à l'intérieur de la zone de conflit de p. Ceci implique que tous les d-simplexes de $\mathrm{Del}(P)$ incidents à σ sont dans la zone de conflit.

Dans le diagramme de Voronoï $\mathrm{Vor}(P)$, ceci se traduit par le fait que tous les sommets de la face σ^* duale de σ sont centres de boules de Delaunay dont l'intérieur contient p. Autrement dit, ces sommets sont plus proches de p que de leurs plus proches voisins dans P. Dit encore autrement, ces sommets se trouvent à l'intérieur de la cellule de p dans le diagramme de Voronoï mis à jour $\mathrm{Vor}(P \cup \{p\})$. Comme cette cellule est convexe, elle contient l'enveloppe convexe des sommets de la face σ^*, qui n'est autre que la face elle-même puisque par hypothèse σ est à l'intérieur de la zone de conflit $C(p)$ et donc n'est pas sur le bord de l'enveloppe convexe de P. □

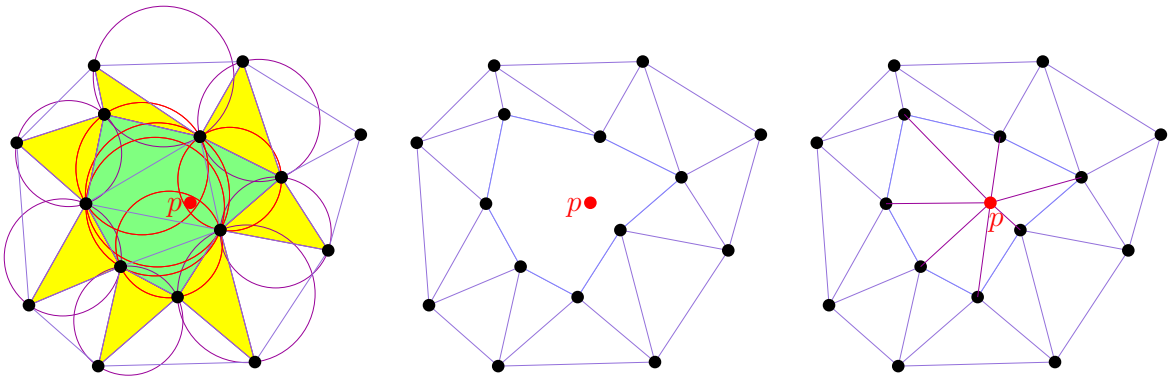


Figure 2.3 – *Procédure d'insertion d'un sommet. À gauche : détection de la zone de conflit $C(p)$. L'intérieur de la zone est marqué en vert, tandis que les triangles bordant la zone à l'extérieur sont marqués en jaune. Au centre : l'intérieur de la zone de conflit est supprimé. À droite : le trou est étoilé par rapport au nouveau sommet p. (Images d'Olivier Devillers)*

Les lemmes 2.14 et 2.15 suggèrent une procédure simple pour déterminer l'ensemble S des simplexes de $\mathrm{Del}(P)$ à supprimer lors de l'insertion de p :

1. Localiser p dans la triangulation, i.e. trouver un d-simplexe σ_p de $\mathrm{Del}(P)$ qui contient p.
2. En partant de σ_p, effectuer un parcours en largeur (dfs) dans la triangulation, et pour chaque d-simplexe σ rencontré tester si ce simplexe est dans $C(p)$, ce qui se fait simplement en testant si p est à l'intérieur de la boule circonscrite à σ. Dans l'affirmative, ajouter σ à S et poursuivre le parcours à partir de σ. À la fin du parcours, ajouter dans S toutes les faces des d-simplexes de S qui sont dans l'intérieur de $C(p)$.

Cette procédure est illustrée dans la figure 2.3. Notez qu'à l'étape 2 on commence par considérer les d-simplexes, pour lesquels le test d'appartenance à la zone de conflit est facile. Ensuite, les simplexes de dimensions plus petites sont ajoutés en tant que faces de d-simplexes. D'après le lemme 2.15, l'ensemble S construit par cette procédure coïncide avec l'ensemble des simplexes de $\mathrm{Del}(P)$ à supprimer lors de l'insertion de p. Reste maintenant à ajouter les simplexes de Delaunay créés lors de l'insertion. Pour simplifier cette étape, nous allons nous appuyer sur la propriété suivante :

Lemme 2.16. *Les simplexes de Delaunay créés lors de l'insertion de p dans la triangulation sont tous incidents à p.*

Démonstration. L'insertion de p dans P ne fait que rétrécir les cellules de Voronoï des autres points de P. En conséquence, si $p_0, \cdots, p_k \in P$ ont des cellules avec une intersection commune non vide après insertion de p, alors leurs cellules avant insertion de p avaient déjà une intersection commune non vide, et donc le simplexe $(p_0, \cdots, p_k)$ était déjà dans $\mathrm{Del}(P)$. □

Quels sont donc ces nouveaux simplexes de Delaunay ? Après suppression des simplexes de l'intérieur de $C(p)$, il reste un "trou" dans la triangulation. D'après le lemma 2.14, ce trou est

étoilé par rapport à p. La seule manière de le combler par des simplexes incidents à p tout en maintenant une triangulation est la suivante (voir la figure 2.3 (droite) pour une illustration) :

3. *Étoiler* par rapport à p le trou laissé par la suppression de l'intérieur de la zone de conflit $C(p)$. Plus précisément, créer tous les simplexes de sommet p et de base un simplexe du bord de $C(p)$.

Les simplexes créés à cette étape sont forcément de Delaunay puisqu'il n'y a pas d'autre moyen de trianguler la zone de conflit avec des simplexes incidents à p. Ainsi, la triangulation de Delaunay est mise à jour lors de l'insertion d'un nouveau sommet. La mise à jour lors de la suppression d'un sommet s'effectue d'une manière similaire.

Notez que nous avons volontairement laissé de côté le cas où p se trouve à l'extérieur de l'enveloppe convexe $\text{conv}(P)$. Dans ce cas, ce qu'on appelle *intérieur* de la zone de conflit doit comprendre non seulement les simplexes formant l'intérieur du complexe $C(p)$ muni de la topologie usuelle de $\mathbb{R}^d$, mais également l'ensemble des simplexes formant l'intérieur du complexe $C(p) \cap \partial\text{conv}(P)$ muni de la topologie induite sur le bord $\partial\text{conv}(P)$ de l'enveloppe convexe de P. Voir la figure 2.4 (gauche) pour une illustration. Les étapes de localisation et de parcours dans la triangulation (étapes 1 et 2 ci-dessus) doivent être adaptées en conséquence, afin que certains simplexes du bord de $\text{conv}(P)$ soient supprimés et qu'ainsi l'enveloppe convexe soit mise à jour.

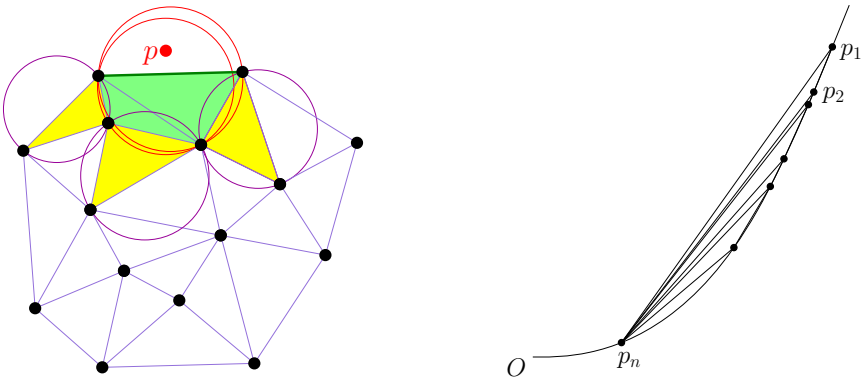


FIGURE 2.4 – *Gauche : zone de conflit d'un point p situé hors de l'enveloppe convexe. L'intérieur de la zone est marqué en vert (foncé pour la partie localisée sur le bord de l'enveloppe convexe). Droite : cas de figure où l'algorithme incrémental crée un nombre quadratique de simplexes au total, car chaque nouveau point inséré est adjacent à tous ses prédécesseurs dans la triangulation de Delaunay.*

Complexité. L'étape 2 ci-dessus se réduit à un simple parcours de l'intérieur de la zone de conflit $C(p)$ et a donc une complexité linéaire en le nombre de simplexes formant cet intérieur. En notant d_p leur nombre, on a que la complexité de l'étape 2 est $O(d_p)$. Cette analyse néglige le coût du test géométrique d'appartenance de p à la boule circonscrite à un d-simplexe, qui au final ne s'avère pas modifier l'analyse asymptotique de complexité et sera donc considéré comme une constante pour simplifier.

L'étape 3 se réduit à un simple parcours du bord de la zone de conflit $C(p)$, dont la taille est exactement le nombre c_p de simplexes de Delaunay créés lors de l'insertion de p. Ainsi, la complexité de l'étape 3 est $O(c_p)$.

Reste l'étape 1, dont la complexité dépend de l'implémentation utilisée pour la localisation de p dans $\text{Del}(P)$. Si la localisation par marche aléatoire décrite dans la section 1.4 est adoptée, alors la complexité moyenne de l'étape 1 sera $O(\sqrt{n})$ dans le plan, et inconnue en dimension

supérieure. Si la localisation par hiérarchie de triangulations proposée par Kirkpatrick (voir la section 5.1.2 dans le chapitre 5) est adoptée, alors la complexité de la localisation sera $O(\log n)$ en toutes dimensions, à condition que la hiérarchie puisse être construite, ce qui pour l'instant n'est assuré que dans le plan. Une troisième approche, proposée par Devillers [Dev02], combine les deux approches ci-dessus. L'idée est de construire une hiérarchie de triangulations aléatoire en insérant tout nouveau point au bas de la hiérarchie et en le remontant avec une certaine probabilité au niveau immédiatement supérieur dans la hiérarchie, et ainsi de suite. Lors de la localisation, chaque niveau de la hiérarchie est exploré à l'aide d'une marche aléatoire, dont le simplexe d'arrivée sert de point de départ à la marche aléatoire effectuée au niveau immédiatement en-dessous. La complexité moyenne d'une requête est alors $O(\log n)$ en toutes dimensions.

La complexité totale de la construction incrémentale de la triangulation avec n points $p_1, \cdots, p_n$ est donc $O(n \log n + \sum_{i=1}^{n}(d_{p_i} + c_{p_i}))$. Pour borner c_{p_i}, il suffit de remarquer que le bord de la zone de conflit $C(p_i)$ est homéomorphe à une $(d-1)$-sphère puisque $C(p_i)$ est étoilée par rapport à p_i. Le bord possède donc $O(n^{\lceil \frac{d-1}{2} \rceil})$ simplexes, d'après la variante du théorème de la borne supérieure démontrée par Stanley [Sta75] et décrite dans notre note culturelle 2.13. On en déduit que $c_{p_i} = O(n^{\lceil \frac{d-1}{2} \rceil})$ à chaque itération i, ce qui donne $\sum_{i=1}^{n} c_{p_i} = O(n^{\lceil \frac{d+1}{2} \rceil})$ au total. Maintenant, comme chaque simplexe détruit doit d'abord être créé, on a $\sum_{i=1}^{n} d_{p_i} \leq \sum_{i=1}^{n} c_{p_i} = O(n^{\lceil \frac{d+1}{2} \rceil})$. On en conclut la borne suivante sur la complexité totale de l'algorithme incrémental :

Théorème 2.17. *Le temps de calcul de l'algorithme incrémental sur un nuage de n points dans $\mathbb{R}^d$ est $\Theta(n \log n + n^{\lceil \frac{d+1}{2} \rceil})$.*

L'algorithme incrémental n'a donc une complexité optimale qu'en dimension impaire, contrairement à la méthode basée sur le relèvement dans l'espace des sphères qui est optimale en toutes dimensions. Toutefois, son énorme avantage est d'être *online*, c'est-à-dire que l'ensemble des sommets de la triangulation n'a pas besoin d'être connu à l'avance, et la structure peut être mise à jour alors que d'autres sommets sont fournis.

Voici un exemple illustrant la non-optimalité de l'approche incrémentale en dimension paire : supposons que $d = 2$ et que les n points de P sont échantillonnés le long de la demi-parabole $\{(x, x^2) \mid x \geq 0\}$. Si les points de P sont insérés par ordre décroissant d'abcisses, alors à chaque insertion le nouveau sommet est voisin de tous les autres dans la triangulation (voir la figure 2.4 (droite)), ce qui fait que le coût total de la construction est quadratique en n alors que la taille du résultat est seulement linéaire en n. Une solution simple pour éviter ce problème consiste à permuter aléatoirement l'ordre d'insertion des points de P dans la triangulation avant de démarrer l'algorithme. Ainsi on obtient une complexité moyenne optimale — voir [Dev02] pour les détails. Toutefois, cette astuce ne peut être utilisée que quand tous les points de P sont connus à l'avance, ce qui n'est pas le cas dans certains scénarios comme ceux de la reconstruction de courbes ou surfaces en ligne. Des stratégies de regroupement des points fournis dans le flux d'entrée ont été proposées pour de tels scénarios, avec plus ou moins de succès, mais le sujet reste essentiellement ouvert.

2.4 Extensions et notes bibliographiques

L'usage informel des diagrammes de Voronoï remonte à Descartes, mais le concept n'a été formalisé qu'au début du 20-ème siècle par Gregory Voronoï. Le lien avec les triangulations de Delaunay a été établi par Boris Delaunay dans les années 1930. Depuis, de nombreuses variantes ont été proposées : ces dernières jouent essentiellement sur la métrique utilisée pour définir les

diagrammes. Ici nous nous sommes restreints à la distance euclidienne, mais d'autres distances peuvent êtres considérées, comme par exemple celles associées aux normes l_p pour $p \in [0, +\infty]$. Toutes ne donnent pas des diagrammes affines (i.e. dont les faces sont des sous-ensembles convexes de sous-espaces affines), bien au contraire. En conséquence, leurs diagrammes ne sont pas forcément duaux de triangulations. Parmi les nouveautés les plus récentes dans cette veine, on compte en particulier les travaux de Shewchuck et Labelle sur les diagrammes de Voronoï anisotropes dans le plan [LS03], revus et généralisés par Boissonnat, Wormser et Yvinec en toutes dimensions [BWY08].

Parmi les nombreuses métriques à notre disposition, quelles sont celles qui donnent des diagrammes affines ? La réponse a éte fournie dans les années 1980 par F. Aurenhammer [Aur87], qui a introduit la métrique dite *de puissance*, dans laquelle chaque point p du nuage P se voit attribuer un poids ω_p, et la distance d'un point $x \in \mathbb{R}^d$ à p devient $\|x - p\|^2 - \omega_p^2$. L'hyperplan bisecteur entre (p, ω_p) et (q, ω_q) est maintenant le lieu des points $x \in \mathbb{R}^d$ satisfaisant l'équation $\|x - p\|^2 - \omega_p^2 = \|x - q\|^2 - \omega^2$, qui après simplification donne une équation affine d'inconnues les coordonnées de x. Les diagrammes obtenus pour la métrique de puissance sont appelés simplement diagrammes de puissance, ou encore diagrammes de Laguerre. Le résultat d'Aurenhammer dit que tout diagramme affine dans $\mathbb{R}^d$ peut s'exprimer comme un diagramme de puissance, pour un nuage de points et un jeu de poids bien définis. Notez enfin que, les diagrammes de puissance étant affines, leurs complexes duaux sont des triangulations, appelées *triangulations régulières*. Ces dernières jouent un rôle très important en génération de maillages ainsi qu'en reconstruction, comme nous l'évoquerons au chapitre 4.

Les applications des diagrammes de Voronoï et des triangulations de Delaunay sont nombreuses et variées, et ne se limitent certainement pas à celles mentionnées ci-dessus. Nous invitons le lecteur à se référer aux chapitres 17 et 18 de [BY02] pour de plus amples informations.

Bibliographie

[Aur87] F Aurenhammer. Power diagrams : properties, algorithms and applications. *SIAM J. Comput.*, 16(1) :78–96, 1987.

[BWY08] J.-D. Boissonnat, C. Wormser, and M. Yvinec. Locally uniform anisotropic meshing. In *Proc. ACM Sympos. on Computational Geometry*, pages 270–277, 2008.

[BY02] J.-D. Boissonnat and M. Yvinec. *Géométrie Algorithmique*. Dunod, collection Ediscience, 2002.

[dBvKOS00] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry : Algorithms and Applications*. Springer-Verlag, Berlin Heidelberg, 2000. 2nd Edition.

[Dev02] O. Devillers. The Delaunay Hierarchy. *International Journal of Foundations of Computer Science*, 13 :163–180, 2002.

[GS85] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph.*, 4(2) :74–123, 1985.

[LS03] F. Labelle and J. R. Shewchuk. Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *SCG '03 : Proceedings of the nineteenth annual symposium on Computational geometry*, pages 191–200, 2003.

[RS85] B. L. Rothschild and E. G. Straus. On triangulations of the convex hull of n points. *Combinatorica*, 5(2) :167–179, 1985.

[Sta75] R. Stanley. The Upper Bound Conjecture and Cohen-Macaulay rings. *Studies in Applied Math.*, 54 :135–142, 1975.

Chapitre 3

Aspects géométriques des graphes (planaires)

3.1 Introduction et motivations

Les graphes planaires ont toujours constitué un objet d'étude fascinant, bien avant la naissance de l'informatique. Leur intérêt repose sans doute sur la possibilité de jeter un lien entre mathématiques discrètes et mathématiques du continu. Par exemple, les graphes plongés (ou cartes) jouent un rôle crucial dans la preuve du théorème de classification des surfaces surfaces compactes. Ou encore, on peut mentionner le théorème de Steinitz qui établit un lien entre graphes et géométrie : les graphes (squelettes) des polyèdres convexes correspondent exactement aux graphes planaires 3-connexes.

L'étude des graphes planaires (et des graphes plongés sur des surfaces) a été aussi motivé par de nombreuses applications. Un exemple est donné par la conception de circuits intégrés à très grande échelle (*VLSI*) où l'on vise à dessiner un réseau électronique de manière à ce que les connections entre les composantes du réseau ne se croisent pas (ou qu'elles croisent avec le nombre minimal d'intersections). Ou encore dans d'autres domaines d'application, tels que la modélisation géométrique ou la simulation, où les maillages (et donc les graphes plongés sur des surfaces) fournissent une représentation des objets géométriques faciles à manipuler.

Il existe une vaste littérature sur les propriétés des graphes planaires, et il serait inutile (et impossible) de fournir ici une présentation exhaustive de ce domaine. Ici on vise plutôt à donner un aperçu général des propriétés et applications de graphes planaires. En particulier nous visons à présenter quelques une des caractérisations récentes de la notion de planarité d'un graphe. On verra comme la planarité peut s'exprimer de différentes façons, qui font intervenir parfois des propriétés géométriques ou topologiques, ou parfois des propriétés algébriques et combinatoires des graphes planaires.

3.2 Graphes : le point de vue combinatoire

3.2.1 Cartes (planaires ou plongées sur des surfaces)

Jusqu'à présent on a parlé de cartes , maillages et polyèdres en s'appuyant sur la définition fournie au Chapitre 1. Rappelons qu'une *carte topologique* est définie par un plongement cellulaire d'un graphe dans le plan ou sur une surface : le graphe est dessiné sans croisements, de manière à ce que les faces soient simplement connexes. De plus, les cartes sont considérées à homéomorphismes près.

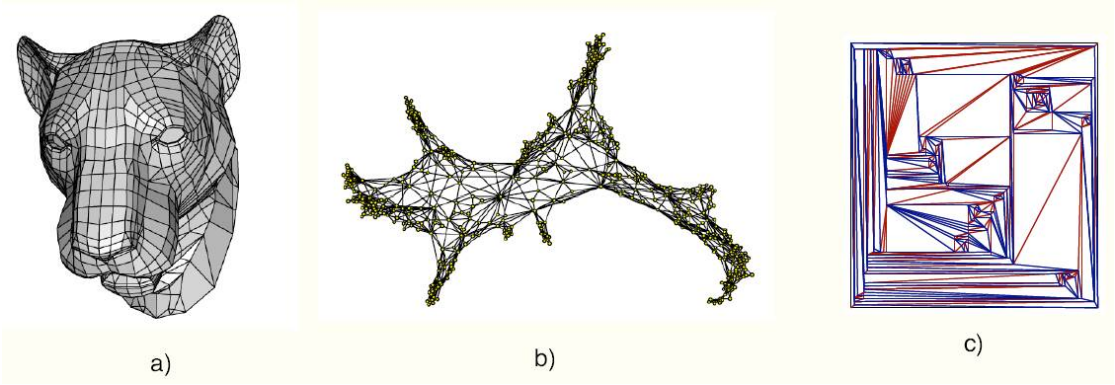


FIGURE 3.1 – *Un maillage surfacique homéomorphe à une sphère (a), une surface discrète aléatoire plongée dans* $\mathbb{R}^3$ *(b), et un dessin d'un graphe planaire (c). (images d'Eric Fusy)*

La condition que les faces soient homéomorphes à un disque topologique se traduit, dans le cas planaire, dans le fait que le graphe doit être connexe : cela dépend d'une variante du célèbre lemme de Jourdan [1], qui s'exprime comme suit :

Lemme 3.1. *Tout dessin planaire d'un cycle* C_i *(pour* $i \geq 3$*) possède exactement deux faces.*

Cartes combinatoires. Il est aussi possible de définir la notion de carte de manière purement combinatoire, à l'aide de la définition suivante (illustrée par la Figure 3.2) :

Définition 3.2. *Une* carte combinatoire $\mathcal{C}$ *est un triplet* (α, σ, ϕ) *constitué de trois permutations sur l'ensemble* H *des* brins *(ou demi-arêtes non orientées) ayant taille* $2n$ *:*
- α *est une involution sans point fixe ;*
- $\alpha\sigma\phi$ *est l'identité ;*
- *le groupe engendré par* α*,* σ *et* ϕ *agit transitivement sur* H*.*

De manière plus intuitive, on peut interpréter les cycles des permutations α, σ et ϕ comme les arêtes, sommets et faces de la carte $\mathcal{C}$. Par exemple, étant donnés une demi-arête (ou brin) d'indice i, $\alpha(i)$ fournit l'indice de la demi-arête opposée. Ou encore, la permutation σ permet de lister (en sens horaire) les demi-arêtes incidentes à un sommet donné (comme illustré par l'exemple de la Figure 3.2).

La première condition de la définition précédente se traduit alors, en utilisant le langage des structure de données vues au Chapitre 1, avec la relation :
- `e.opposite.opposite` = `e` (`e` étant une demi-arête).

L'*enracinement* d'une carte consiste à marquer et orienter une arête de telle sorte qu'elle ait la face infinie (ou externe) à sa droite : une telle carte est dite *enracinée*.

Les *relations d'incidences* décrivent la manière dont les éléments d'une carte sont reliés : un sommet v et une arête e sont incident lorsque v est une extrémité de e ; une face f est incidente à une arête e (resp. un sommet v) si e (resp. v) appartient au bord de f. D'un point de vue combinatoire, une carte est entièrement donnée par les relations d'incidence sommets/aretes/faces et l'arrangement cyclique des arêtes autour des sommets ou des faces. Ainsi, dans le cas des graphes, la seule information disponible concerne les incidences arêtes / sommets ; alors que c'est avec la notion de carte qu'on dispose de l'information topologique concernant, par exemple, les incidences faces / sommets.

1. Dans sa formulation usuelle, le lemme de Jourdan s'exprime de la manière suivante : toute courbe γ fermée simple dans le plan sépare le plan en deux régions disjointes (l'une bornée, et l'autre infinie), dont γ est le bord.

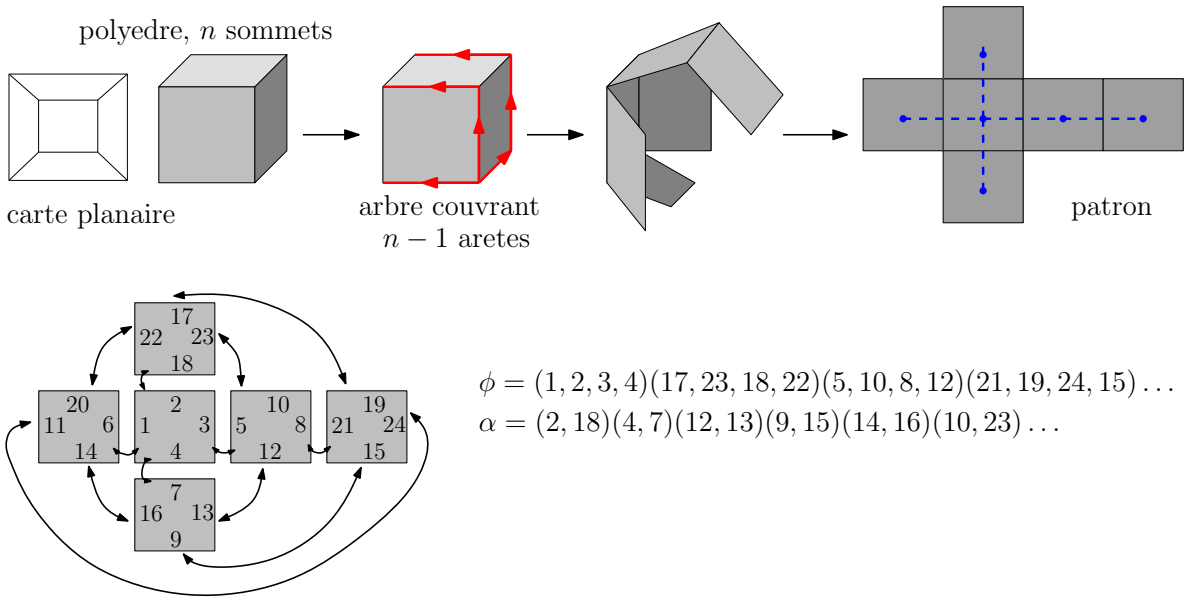


Figure 3.2 – *Les premières images illustrent une revisitation de la formule d'Euler. Un exemple de carte combinatoire (en bas).*

3.2.2 Graphes et 3-connexité.

Le lien entre graphes et cartes se révèle encore plus étroit lorsque des contraintes supplémentaires concernant la connexité du graphe sont imposées. Plus précisément, les triangulations planes 3-connexes (ou autrement dit, les triangulations planes sans arêtes multiples) sont équivalentes aux *graphes plans maximaux* (et aussi aux 2-triangulations de la sphère), comme exprimé par le théorème suivant :

Théorème 3.3 (Whitney, 1933)**.** *Un graphe planaire 3-connexe admet un unique plongement planaire à homéomorphisme et inversion de la sphère près.*

Mais encore plus significatif dans ce contexte se révèle le théorème de Steinitz (1916) unifiant des propriétés des graphes, des complexes de dimension 2 et des surfaces discrètes qui, dans sa formulation moderne en théorie des graphes, s'exprime par :

Théorème 3.4 (Steinitz)**.** *Un graphe G est isomorphe au 1-squelette d'un polyèdre convexe de dimension 3 (3-polytope) si et seulement si G est planaire et 3-connexe.*

Il est facile de rendre compte que le squelette de tout polyèdre convexe correspond à un graphe planaire : comme déjà mentionné au Chapitre 1, il suffit de considérer un point de l'espace assez proche de l'une des faces du polyèdre et de projeter les sommets (et arêtes) sur un plan placé de l'autre coté par rapport au polyèdre. Le dessin qui en résulte est planaire, et de plus toutes les arêtes correspondent à des segments de droites : celle-ci n'est pas un restriction du théorème de Steinitz, en vertu de la propriété suivante :

Théorème 3.5 (Fáry , 1947)**.** *Tout graphe planaire simple G (sans boucle ni arêtes multiples) peut être dessiné dans le plan (sans croisements) de manière à ce que les arêtes de G correspondent à des segments de droites.*

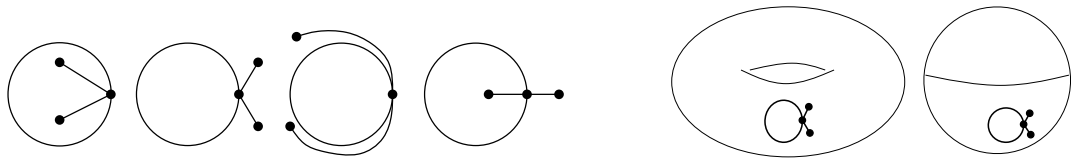


FIGURE 3.3 – *Les premières images montrent quatre plongements d'un même graphe planaire, définissant deux cartes différentes. Les trois premiers plongements, ayant des faces de degré* 1 *et* 5*, représentent la même carte : les deux premiers diffèrent seulement pour le choix de la face infinie, tandis que le troisième peut s'obtenir du deuxième par déformation continue. Le quatrième dessin correspond à une carte différente ayant deux faces de degré* 3*. Les deux dernières images montrent deux dessins du graphe sur un tore et une sphère respectivement : seulement le deuxième définit une carte, puisque sur le tore la face externe n'est pas homéomorphe à un disque ouvert.*

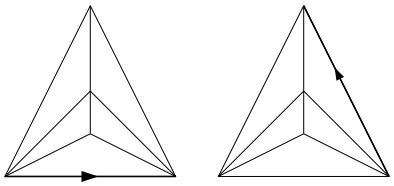


FIGURE 3.4 – *Ces deux premières images montrent le plongement d'une triangulation planaire sans arête multiple : en tant que graphe* 3*-connexe le plongement est essentiellement unique. En tant que cartes enracinés, les deux triangulations sont différentes.*

Les théorèmes mentionnés ci-dessus représentent des résultats fondamentaux de la théorie des graphes : ils font intervenir d'intéressantes propriétés géométriques et algébriques, qui feront l'objet des Sections 3.5.2 et 3.4.

3.3 Deux célèbres caractérisations des graphes planaires

3.3.1 Théorème de Kuratowski

Rappelons la relation d'Euler, qui est une propriété fondamentale des graphes planaires : elle permet déjà de fournir une première caractérisation de la planarité, faisant intervenir juste des arguments de comptage.

Pour tout graphe planaire ayant n sommets, f faces et e arêtes, la relation suivante est satisfaite :

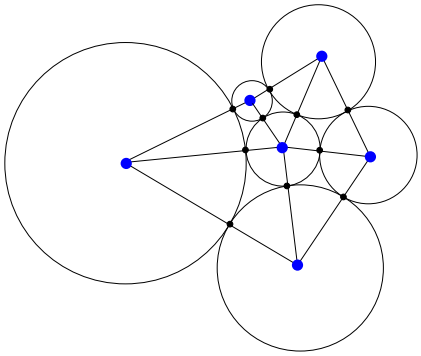
$$n - e + f = 2$$

Cette simple relation nous permet déjà d'avoir une borne supérieure sur le nombre d'arêtes de certaines graphes planaires.

Lemme 3.6. *Soit G un graphe planaire (simple) avec n sommets et e arêtes. Si G ne contient pas de triangle (cycle de longueur* 3*) alors on a $e < 2n - 4$.*

Démonstration. Comme le graphe G ne contient pas de triangle, alors toute face de G est incidente à au moins 4 arêtes. Mais alors, par double comptage des arêtes de G, on a que $4f \leq 2e$ (chaque arête appartient à deux faces). Cette relation, et la formule d'Euler, suffisent pour terminer la preuve. □

(a)
Représentation de Koebe d'un graphe planaire



(b)
Représentation de Polyèdres

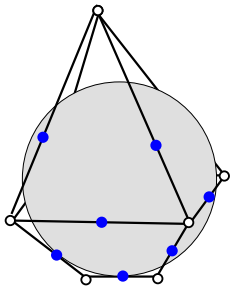


Figure 3.5 – *Ces images illustrent la représentation des graphes planaires à base d'intersections de disques.*

Cette propriété, ainsi que le lemme 1.7 (conséquences aussi de la formule d'Euler) permettent d'établir que certaines graphes ne peuvent pas être planaires.

Corollaire 3.7. *Les graphes K_5 (graphe complet à n sommets) et le graphe $K_{3,3}$ (graphe complet biparti avec 3 sommets dans chaque partition) ne sont pas planaires.*

Démonstration. Considérons le graphe $K_{3,3}$ qui, étant biparti, ne possède pas de cycle de longueur 3 : pour le lemme précédent $K_{3,3}$ peut avoir au plus $e \leq 8$ arêtes. Alors que par construction $K_{3,3}$ possède 9 arêtes. De manière similaire, K_5 devrait avoir au plus $e \leq 3 \cdot 5 - 6 = 9$ arêtes (lemme 1.7) : mais par définition il possède $e = \binom{5}{2} = 10$ arêtes. □

Ce qu'on a prouvé jusqu'à présent suffit pour montrer un sens (facile) du célèbre théorème de Kuratowski.

Théorème 3.8 (Kuratowski)**.** *Un graphe G est planaire si et seulement si G ne contient pas $K_{3,3}$ ni K_5 comme mineurs.*

3.3.2 Théorème de Koebe

Une première caractérisation géométrique des graphes planaires est donnée en termes de graphes d'intersection de disques.

Considérons une collection $\mathcal{D} = \{D_0, \ldots, D_{n-1}\}$ de disques dans $\mathbb{R}^2$ tels que leurs intérieurs ont une intersection vide (une telle collection est aussi connue sous le nom de *disk packing*). On définit le *graphe d'intersection* de $\mathcal{D}$ comme le graphe dont les sommets sont $\{D_0, \ldots, D_{n-1}\}$ et ayant une arête (D_i, D_j) pour tout couple de disques D_i et D_j ayant un point d'intersection commun. Il est facile de s'apercevoir qu'un graphe d'intersection d'une telle famille de disques est planaire et peut se dessiner avec des segments de droites dans le plan (admettant donc un *straight-line planar drawing* : pour tout sommet du graphe, il suffit de choisir comme point du plan le centre du disque correspondant).

Le théorème suivant constitue un résultat majeur de la théorie des graphes et dont on peut obtenir de nombreuses applications [2].

2. La stratégie de preuve fournie ici suit la présentation proposée par V. Koltun.

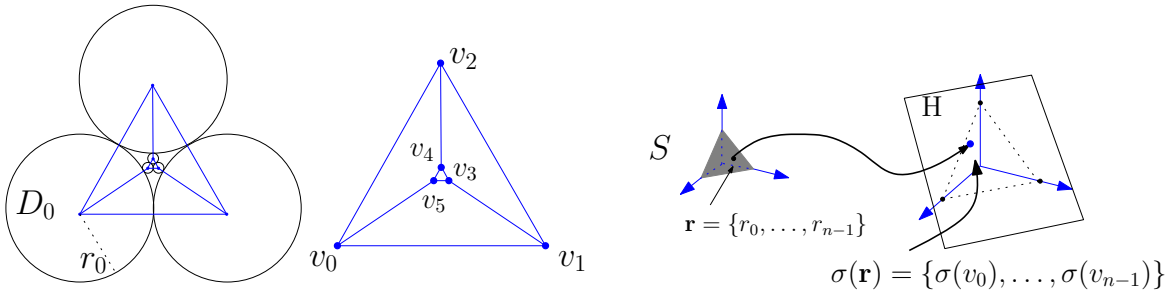


FIGURE 3.6 – *Ces images illustrent la preuve du théorème de Koebe.*

Théorème 3.9 (Koebe-Andreev-Thurston)**.** *Pour tout graphe planaire G à n sommets il existe une collection de disques $\{D_0, \ldots, D_{n-1}\}$ dont le graphe d'intersection est isomorphe à G.*

Une propriété cruciale qu'on va utiliser pour montrer le théorème de Koebe est le fait que, dans un graphe planaire, la somme des angles des triangles adjacents à un sommet interne est toujours égale à 2π. Pour formaliser cette propriété on va d'abord introduite quelques notations.

Considérons un graphe planaire G muni d'une représentation de Koebe ρ : pour tout sommet v_i $(0 \leq i \leq n-1)$ soit r_i le rayon du disque D_i correspondant au sommet v_i. Sans perte de généralité on peut supposer que $\sum_{i=0}^{n-1} r_i = 1$: si ce n'est pas le cas, on peut toujours normaliser en divisant par $R = \sum_i r_i$. Dénotons par $\overrightarrow{r} = \{r_0, r_1, \ldots, r_{n-1}\}$ le vecteur ayant comme composantes les rayons r_i, et soit $\sigma_{\overrightarrow{r}}(v_i)$ l'application qui fait correspondre, à chaque sommet v_i, la somme des angles des faces incidentes à v_i, par rapport à la représentation fournie par les rayons de $\overrightarrow{r}$.

Lemme 3.10. *Etant donnés des rayons $\{r_0, \ldots, r_{n-1}\}$ il existe une représentation de Koebe d'un graphe G, ayant n sommets et face externe (v_0, v_1, v_2), si et seulement si pour tout les sommets internes on a :*

$$\sigma_{\overrightarrow{r}}(v_i) = 2\pi, \qquad \forall i,\ 3 \leq i \leq n-1 \tag{3.1}$$

Compte tenu de cette propriété, et du fait que dans une triangulation planaire à n sommets le nombre de faces est $f = 2n - 4$, la partie restante de cette section sera dédiée à montrer l'existence d'un ensemble de n rayons $\{r_0, r_1, \ldots, r_{n-1}\}$ satisfaisant les conditions suivantes :

$$(\sigma(v_0), \sigma(v_1), \sigma(v_2), \sigma(v_3), \ldots, \sigma(v_{n-1})) = (\frac{2\pi}{3}, \frac{2\pi}{3}, \frac{2\pi}{3}, 2\pi, \ldots, 2\pi) \tag{3.2}$$

$$\sum_{i=0}^{n} \sigma(v_i) = f\pi = (2n-4)\pi \tag{3.3}$$

Considérons l'ensemble $S \subset \mathbb{R}^n$ défini par les vecteurs dont les composantes sont des rayons valides :

$$S := \{\overrightarrow{r} = (r_0, \ldots, r_{n-1}) \,|\, r_i > 0 \text{ et } \sum_i r_i = 1\}$$

et l'hyperplan H formé par tous les points $\overrightarrow{x} \in \mathbb{R}^n$ qui satisfont la deuxième condition ci-dessus :

$$H := \{\overrightarrow{x} = (x_0, \ldots, x_{n-1}) \,|\, x_0 + x_1, + \ldots, + x_{n-1} = (2n-4)\pi\}$$

La preuve du théorème 3.9 découle des propriétés de la fonction $h : S \longrightarrow H$ définie ci-dessous (voir les quatre lemmes qui suivent) :

$$h(\overrightarrow{r}) := (\sigma(v_0), \ldots, \sigma(v_{n-1}))$$

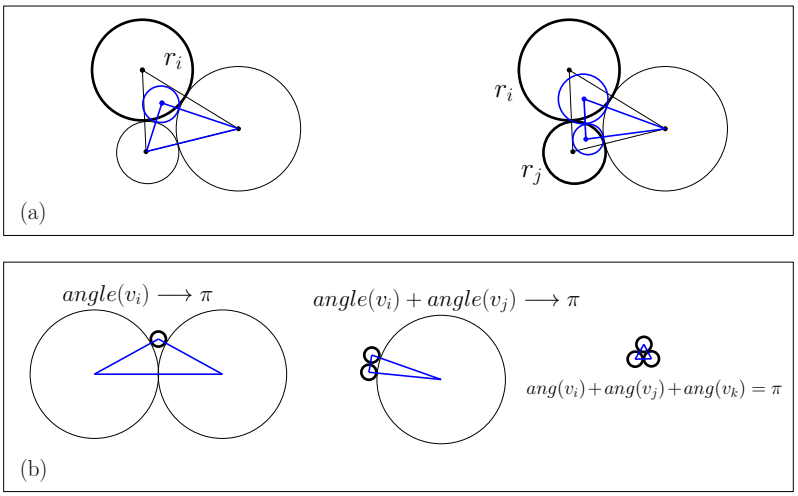


Figure 3.7 – *Ces images illustrent deux étapes de la preuve du théorème de Koebe.*

Lemme 3.11. *L'application $h : S \longrightarrow H$ est injective.*

Démonstration. On considère deux vecteurs $\overrightarrow{r}, \overrightarrow{r}' \in S$, où $\overrightarrow{r} \neq \overrightarrow{r}'$, et on va montrer que $h(\overrightarrow{r}) \neq h(\overrightarrow{r}')$. Puisque $\overrightarrow{r}$ et $\overrightarrow{r}'$ ne coïncident pas alors il existe des indices i tels que $r_i \neq r'_i$. De plus, sans perte de généralité on peut supposer qu'il existe un ensemble d'indices I tel que $r_i < r'_i$ pour tout $i \in I$: par construction $I \neq \emptyset$, et aussi $I \neq \{0, \ldots, n-1\}$ car on sait que $\sum_{i<n} r_i = 1$ et $\sum_{i<n} r'_i = 1$. Considérons trois cercles correspondant à trois sommets v_i, v_j et v_k formant une face triangulaire de G, et dénotons par r_i, r_j et r_k leurs rayons (comme illustré par la Figure 3.7(a)). Si on fait augmenter le rayon r_i, alors que les rayons r_j et r_k diminuent (ou restent inchangés), on remarque que l'angle correspondant au sommet v_i va diminuer. De manière similaire, si on fait augmenter r_i et r_j, alors qu'on diminue (ou on laisse inchangé) la valeur de r_k, on remarque que la somme des deux angles correspondant aux sommets v_i et v_j va diminuer. Cela permet d'établir

$$\sum_{i \in I} \sigma_r > \sum_{i \in I} \sigma_{r'}$$

qui montre que $h(r) \neq h(r')$. □

Lemme 3.12. *Pour tout vecteur $\overrightarrow{r} \in S$, l'image $h(\overrightarrow{r})$ est contenue dans l'intérieur d'un certain polytope $P^* \subset H$, défini par*

$$P^* := \{\mathbf{x} = \{x_0, \ldots, x_{n-1}\} \mid \sum_i x_i = (2n-4)\pi,\ \sum_{i \in I} x_i < |F(I)|\pi, \forall I\}$$

où I est un sous-ensemble de $\{0, \ldots, n-1\}$, et $F(I)$ dénote l'ensemble des faces ayant au moins un sommet incident v_i, avec $i \in I$.

Démonstration. On va montrer que l'image du bord de S par l'application h est contenue dans le bord de P^* : et pour cela on montrera que le vecteur $h(r)$ tends au bord du polytope P^* lorsque le vecteur r tends vers le bord de S. Dénotons par I l'ensemble des indices correspondant aux sommets dont le rayon tend vers 0 (lorsque r s'approche du bord de S). On veut prouver que

$$\sum_{i \in I} \sigma(r_i) \longrightarrow |F(I)|\pi$$

Pour cela, on considère une face $\triangle \in F(I)$ et on étudier comment les angles (internes) correspondant à ses trois sommets varient lorsque les rayons tendent à 0. Comme illustré par la Figure 3.7(b) il y a trois cas à prendre en compte, selon le nombre de disques concernés. Dans un premier cas, un seul rayon tends vers 0, et dans ce cas l'angle correspondant du triangle $\triangle$ tends donc vers π. Lorsque deux rayons tendent au meme temps vers 0 (éventuellement avec différentes vitesses de convergence), on sait la somme des deux angles correspondant va tendre vers π. Et finalement, lorsque tous les trois sommet de $\triangle$ appartiennent à l'ensemble V_I, leur somme reste π. Ce qui preuve la validité de la relation ci-dessus. □

Lemme 3.13. *L'application $h : S \longrightarrow P^*$ est surjective.*

Les détails de la preuve de ce lemme sont omis. Sa validité repose sur une propriété de topologie générale. En effet on dispose d'une fonction h qui est continue et injective, et définie sur S à valeurs dans P^*. De plus, lorsque les vecteurs r tendent au bord de S on vient de montrer que leurs images tendent au bord de P^* : ainsi, un théorème de topologie nous garantit que la fonction est aussi surjective.

Lemme 3.14. *Le polytope P^* contient le vecteur $(\frac{2\pi}{3}, \frac{2\pi}{3}, \frac{2\pi}{3}, 2\pi, \ldots, 2\pi)$.*

Démonstration. Etant donné le vecteur $\mathbf{x} = (\frac{2\pi}{3}, \frac{2\pi}{3}, \frac{2\pi}{3}, 2\pi, \ldots, 2\pi)$, on voit facilement que $\sum x_i = (n-3)2\pi + 3\frac{2\pi}{3} = (2n-4)\pi$. Il nous reste à montrer que $\sum_{i\in I} x_i < |F(I)|\pi$: la stratégie à suivre consiste à prouver d'abord que $|F(I)| > 2|I|$. Considérons le sous-ensemble de sommets V_I (dont les indices des sommets appartiennent à I). On supprime de G tous les sommets dans V_I, ainsi que toutes les aretes incidentes à ces sommets. Ce qu'on obtient ainsi est un graphe $\overline{G}$, contenant $n - |I|$ sommets, et dont le nombre de faces est au plus $2(n - |I|) - 4$ (ceci est due à la formule d'Euler). Et finalement, observons que par définition les faces de $\overline{G}$ n'appartiennent pas à $F(I)$: ce qui montre que $|F(I)| > 2|I|$. Et pour conclure, de cette dernière relation et du fait que $x_i \leq 2\pi$, on trouve

$$\sum_{i\in I} x_i \leq 2\pi|I| < |F(I)|\pi$$

□

Nous avons maintenant tous les ingrédients pour montrer la validité du théorème 3.9.

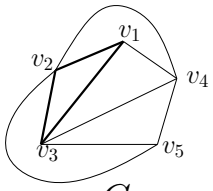
Preuve du théorème de Koebe. Il suffit de remarquer que puisque le vecteur $(\frac{2\pi}{3}, \frac{2\pi}{3}, \frac{2\pi}{3}, 2\pi, \ldots, 2\pi)$ appartient à P^* et la fonction h est surjective, alors il existe des rayons $(r_0, \ldots, r_{n-1})$ tels que

$$h(\overrightarrow{r}) = h(r_0, \ldots, h_{n-1}) := (\sigma(v_0), \ldots, \sigma(v_{n-1})) = (\frac{2\pi}{3}, \frac{2\pi}{3}, \frac{2\pi}{3}, 2\pi, \ldots, 2\pi)$$

ce qui, combiné avec le Lemme 3.10, implique qu'il existe une représentation de Koebe du graphe G. □

Il est maintenant possible d'établir la validité du théorème de Fáry, qui découle directement du théorème 3.9.

Avant de conclure on peut mentionner que la construction de Koebe-Andreev-Thurston conduit aussi à une autre caractérisation des graphes planaires en termes de polyèdres (une forme faible du théorème de Steinitz). En effet on peut montrer que tout graphe planaire 3-connexe correspond au 1-squelette d'un polyèdre convexe en $\mathbb{R}^3$, plongé de manière à ce que ses arêtes soient tangentes à la sphère unité (comme illustré par la Figure 3.9).



$$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad Q_G = \begin{bmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

$$A_G[i,j] = \begin{cases} 1 & v_i \text{ adjacent } v_j \\ 0 & \text{autrement} \end{cases} \qquad Q_G[i,k] = \begin{cases} deg(v_i) & \text{si } i = j \\ -A_G[i,j] & \text{autrement} \end{cases}$$

Figure 3.8 – *Cet exemple illustre la matrice d'adjacence et la matrice laplacienne d'un graphe.*

3.4 Théorie algébrique des graphes

Dans cette section nous allons explorer les propriétés algébriques des graphes, pouvant s'exprimer en termes de valeurs et vecteurs propres de certaines matrices associées aux graphes [3].

On commence par fournir la définition de quelques matrices jouant un rôle important dans le domaine de la représentation et dessin planaire des graphes (ainsi que dans d'autres domaines d'application, tels que la parametrization de maillages).

Définition 3.15. *Étant donné un graphe orienté $\mathcal{G}$ à n sommets, sa* matrice d'adjacence $A(\mathcal{G})$ *est une matrice de taille $n \times n$ telle que $A[i,j]$ est le nombre d'arêtes dirigées du sommet v_i au sommet v_j. Pour un graphe non orienté, la matrice $A(\mathcal{G})$ est définie par*

$$A[i,j] = \begin{cases} 0 & \textit{si } u = v, \textit{ ou } u \textit{ et } v \textit{ ne sont pas adjacents} \\ 1 & \textit{si } u \textit{ est adjacent à } v \end{cases}$$

Alors que la matrice d'adjacence décrit le relations de voisinage entre sommets, une autre matrice (appelée d'*incidence*) décrit les relations d'incidence aretes/sommets.

Définition 3.16. *Étant donné un graphe (non orienté) $\mathcal{G}$ à n sommets et e arêtes, la* matrice d'incidence $D(\mathcal{G})$ *est une matrice à valeurs dans $\{0,1\}$, ayant n lignes et e colonnes définie comme suit :*

$$D[i,k] = \begin{cases} 1 & \textit{si le sommet } v_i \textit{ est incident à l'arete } e_k \\ 0 & \textit{autrement} \end{cases}$$

Pour un graphe orienté, la matrice est définie à valeurs dans $\{-1,0,1\}$ de manière similaire :

$$D[i,k] = \begin{cases} 1 & \textit{si le sommet } v_i \textit{ est la destination de l'arete } e_k \\ -1 & \textit{si le sommet } v_i \textit{ est l'origine de l'arete } e_k \\ 0 & \textit{autrement} \end{cases}$$

Et enfin, on peut introduire la *matrice laplacienne*, pouvant se définir en termes de la matrice d'adjacence, et dont les propriétés spectrales ont joué un rôle important dans la conception d'algorithmes de représentations de graphes.

Définition 3.17. *Étant donné un graphe (non orienté) $\mathcal{G}$ à n sommets, la* matrice laplacienne $Q(\mathcal{G})$ *est de taille $n \times n$ et définie par :*

$$Q[i,j] = \begin{cases} \textit{degré}(v_i) & \textit{si } i = j \\ -A[i,j] & \textit{autrement} \end{cases}$$

3. Cette section est basée sur l'exposition des propriétés spectrales des graphes donnée dans le texte de Godsil et Royle [GR01]

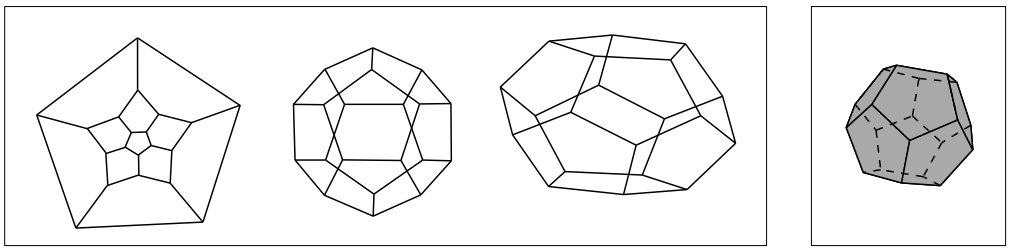


FIGURE 3.9 – Exemples de représentations du graphe correspondannt au squelette d'un dodecaèdre. Les premières trois représentations (en 2D), sont données par la méthode barycentique de Tutte (dessin sans croisement), la méthode spectrale, et la méthode dite *force directed drawing*. La représentation de droite (en 3D) a été obtenue à l'aide de la méthode spectrale.

Une propriété importante est que la matrice laplacienne Q ne dépend pas de l'orientation d'un graphe : cela peut se déduire de la relation suivante, qui fournit une définition alternative de la matrice laplacienne (D^T dénote la matrice transposée de D) :

$$Q(\mathcal{G}) = DD^T$$

Une autre propriété fondamentale, qui sera utilisée dans la suite, permet d'exprimer le calcul du nombre d'arbres couvrants $\tau(G)$ d'un graphe G, à partir du déterminant de sa matrice laplacienne. On va dénoter par $Q[i]$ la sous-matrice de Q qu'on obtient en supprimant la i-ème colonne et i-ème ligne (qui correspondent au sommet v_i du graphe G). Dans la suite on va considérer des graphes pouvant contenir des arêtes multiples, et on va donc considérer une définition légèrement différente de matrice laplacienne. Soit e_{ij} le nombre d'arêtes existantes entre les sommets i et j : alors $Q[i,j]$ vaut $-e_{ij}$ si $i \neq j$, et $deg(v_i)$ autrement. On peut maintenant établir le lemme suivant :

Lemme 3.18. *Soit G un graphe à n sommets, muni de matrice laplacienne Q. Alors, le nombre d'arbres couvrants de G est*

$$\tau(G) = det(Q[i])$$

pour un indice positif i quelconque ($i \leq n$).

3.5 Dessin de graphes planaires

3.5.1 Représentation d'un graphe dans $\mathbb{R}^d$

On commence par définir la notion de *représentation d'un graphe*. Une représentation d'un graphe $\mathcal{G}$ dans $\mathbb{R}^d$ est une application

$$\rho : V(\mathcal{G}) \longmapsto \mathbb{R}^d$$

telle que l'image $\rho(v)$ de tout sommet v de G est un point de $\mathbb{R}^d$. Une représentation ρ est *équilibrée* si la relation suivante est satisfaite :

$$\sum_{v_i \in V} \rho(v_i) = 0$$

ce qui revient à demander que le centre de masse des points dans $\mathbb{R}^d$ (images des sommets de $\mathcal{G}$) est l'origine.

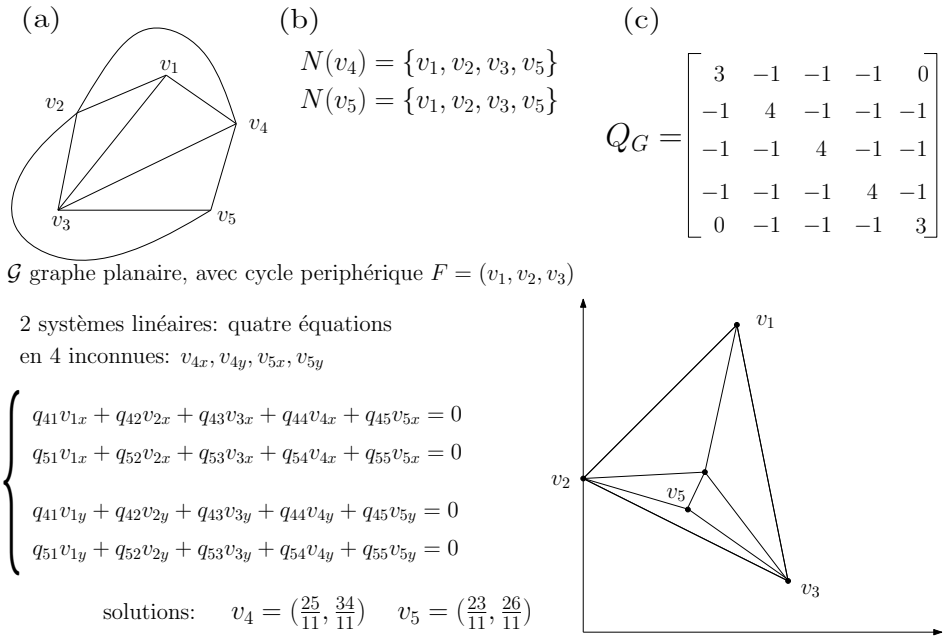


Figure 3.10 – *Ces images illustrent avec un exemple la méthode barycentrique de Tutte, basée sur la résolution de systèmes d'équations linéaires.*

3.5.2 Représentations convexes et méthode de Tutte

Définition 3.19. *Un graphe planaire G admet une* représentation convexe *(ou* dessin convexe*) s'il peut être dessiné de manière à ce que toutes les faces du plongement correspondant soient des polygones convexes.*

Il est facile de voir que tout graphe planaire n'admet pas une telle représentation : par exemple, il faut que le graphe soient au moins 2-connexe.

Si l'on considère un graphe 3-connexe G, le théorème de Steinitz nous dits qu'un tel graphe est isomorphe au 1-squelette d'un polytope de dimension 3. On peut donc espérer qu'avec le choix d'une bonne projection $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$, la projection du 1-squelette de P donne un plongement planaire où chaque cellule est définie par un polygone convexe : le problème revient donc à trouver une telle projection.

Une solution élégante et particulièrement simple à mettre en place, a été proposée par Tutte [Tut63]. L'avantage (qui se révèle un gros désavantage, au même temps) de la méthode de Tutte est que le calcul d'une représentation convexe se réduit à la résolution d'un système de $O(n)$ équations linéaires. Elle est facile et rapide à mettre en place, mais sa complexité peut être assez élevée, comparée à d'autres méthodes : elle nécessite un temps $O(n^3)$ si on applique simplement la méthode de Gauss, et on peut arriver jusqu'à un temps $O(n^{3/2})$ avec des méthodes de résolution plus sophistiquées.

Théorème 3.20 (Tutte [Tut60, Tut63])**.** *Tout graphe planaire 3-connexe G à n sommets admet une représentation convexe. De plus, il est possible de calculer une telle représentation en temps $O(n^{3/2})$.*

Aperçu de la méthode. La méthode de Tutte exploite certaines propriétés des coordonnées barycentriques. On procède de la manière suivante :

— on fixe un ensemble F de sommets (appelé *cycle periphérique*) définissant la face externe du plongement ;
— on choisit un polygone convexe $P = (p_1, \ldots, p_k)$ (où k est la taille du cycle F) : ses extrémités correspondent aux sommets du cycle F ;
— à toute arête interne $e = (v_i, v_j)$ on associe une valeur positive w_{ij} (son poids) de manière à ce que, pour tout sommet v_i, on ait :

$$\sum_{j \in N(v_i)} w_{ij} = 1$$

où $N(v_i)$ dénote l'ensemble des sommets adjacents au sommet v_i ;
— w_{ij} est choisi égal à 0 pour les autres arêtes ;
— on définit une matrice $W = [w_{ij}]$ de taille $(n-k) \times (n-k)$, et deux vecteurs colonne $\overrightarrow{b_x}$ et $\overrightarrow{b_y}$ de taille $n-k$ représentant les coordonnées des points du polygone :

$$\begin{cases} \overrightarrow{b_x}[i] = 0 & \text{(si } v_i \text{ n'est pas adjacent a } F\text{)} \\ \overrightarrow{b_x}[i] = \sum_{j \in N(i)} \frac{x_i}{deg(v_i)} & \text{autrement} \end{cases}$$

— il ne reste qu'à résoudre les deux systèmes d'équations linéaires (par rapport aux vecteurs $\overrightarrow{x}$ et $\overrightarrow{y}$) :

$$(I - W)\overrightarrow{x} = \overrightarrow{b_x},\ (I - W)\overrightarrow{y} = \overrightarrow{b_y}$$

— la représentation barycentrique de G est alors donnée par :

$$\rho(v_i) = (x_i, y_i)$$

Validité de la méthode. Premièrement, il faut observer que les systèmes d'équations ci-dessus admettent une solution (la matrice est inversible) : la preuve sera esquissée dans la suite avec un formalisme légèrement différent.

Et finalement, la validité de ce procédé de dessin repose sur un résultat fondamental, qui s'avère crucial dans plusieurs situations différentes (voir Section 3.5.4) : à toute représentation barycentrique d'un graphe planaire G correspond un dessin sans croisements. Ce qui est exprimé par le lemme suivant :

Lemme 3.21 (Tutte [Tut60])**.** *Soit $\mathcal{G}$ un graphe planaire 3-connexe (simple) à n sommets, muni d'un cycle F définissant la face externe. Considérons un polygone convexe P et une représentation $\rho : \mathcal{G} \longmapsto \mathbb{R}^2$, telle que $\rho(F) = P$. Alors les points $\rho(v_i)$ (pour $i = 1 \ldots n$) définissent une triangulation planaire (les intérieurs des triangles sont deux à deux disjoints) si et seulement si pour tout sommet $u \in V(G)$ le point $\rho(u)$ peut s'exprimer comme une combinaison convexe des points images des voisins de u dans G.*

Ceci montre aussi que la méthode barycentrique permet d'obtenir tous les dessins planaires valides (sans croisements) possibles d'un graphe donné, une fois qu'un polygone convexe définissant la face externe a été choisi.

Méthode originale de Tutte Ce qu'on a esquissé auparavant représente une méthode de dessin assez générale : le choix des poids w_{ij} étant "arbitraire".

Dans la version originale, Tutte proposait de choisir

$$w_{ij} = \frac{1}{degré(v_i)} \quad \text{pour toutes les aretes } (v_i, v_j)$$

ce qui revient à imposer que tout sommet interne corresponde au centre de masse de ses voisins (les poids sont tous égaux).

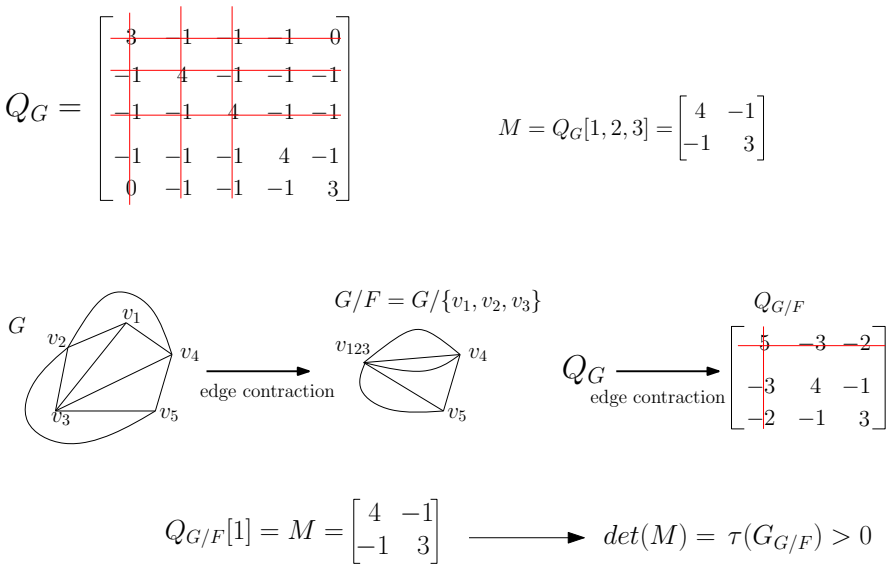


Figure 3.11 – *Ces images illustrent la preuve du théorème barycentrique de Tutte. La méthode barycentrique repose sur la solution de systèmes d'équations linéaires, et fait intervenir les propriétés de la matrice laplacienne Q.*

Désavantages de la méthode barycentrique de Tutte Bien qu'élégante et facile à mettre en place, la méthode de dessin décrite par Tutte possède quelques désavantages. Premièrement, l'algorithme requiert que le graphe soit 3-connexe. Du point de vue de l'implantation, une telle méthode repose sur la résolution de systèmes linéaires, ce qui comporte une efficacité dans la pratique pour des graphes avec au plus une centaines de sommets. Si on considère la complexité des coordonnées des points, en terme du nombre de bits, on observe aussi que le dessin qu'on obtient a une taille exponentielle (par rapport au nombre de sommets). Il existe d'autres algorithmes de dessin qui résolvent de manière élégante ces deux problèmes (voir Section 3.5.4).

Et pour terminer, les propriétés géométriques du dessin ne sont pas toujours satisfaisantes du "point de vue esthétique".

3.5.2.1 Matrice laplacienne et méthode de Tutte

Compte tenu des propriétés mentionnées ci-dessus, on peut réformuler la méthode de Tutte pour le calcul d'un dessin planaire d'une graphe, en terme de matrices laplaciennes.

Lemme 3.22. *Soit G un graphe connexe et F un cycle de G tel que $G \setminus F$ est connexe. Alors, étant donnée une représentation $\rho_F : F \longmapsto \mathbb{R}^d$, il existe une seule application $\rho : G \longmapsto \mathbb{R}^d$ qui satisfait les conditions suivantes :*

- *ρ* étend *ρ_F, c'est à dire : $\rho(v) = \rho_F(v)$, pour tout $v \in F$;*
- *ρ est barycentrique relativement à F.*

Démonstration. Dénotons par $v_1, \ldots, v_n$ les sommets du graphe G, et supposons, sans perte de généralité, que le cycle F est défini par les premiers k sommets de G : $F = \{v_1, \ldots, v_k\}$.

Par définition, l'image de tout sommet interne v_i par ρ doit pouvoir s'exprimer comme combinaison barycentrique des images de ses voisins :

$$\rho(v_i) = \sum_{j \in N(i)} w_j \rho(v_j) = \sum_{j \in N(i)} \frac{1}{deg(v_i)} \rho(v_j) \tag{3.4}$$

En réécrivant $\rho(v_i) - \sum_{j \in N(i)} \frac{1}{deg(v_i)} \rho(v_j) = 0$, on obtient des systèmes d'équations linéaires

$$\begin{cases} (I - W)\overrightarrow{x} = \overrightarrow{b_x} \\ (I - W)\overrightarrow{y} = \overrightarrow{b_y} \end{cases} \tag{3.5}$$

déjà rencontrés au début de la section 3.5.2.

Ce qui est intéressant est la possibilité d'exprimer les solutions d'un tel système de manière à faire intervenir la matrice laplacienne de G. Observons que les équations 3.4 sont équivalentes aux relations suivantes ($deg(v_i) > 0$ pour tout sommet v_i, car le graphe est connexe) :

$$\rho(v_i) deg(v_i) - \sum_{j \in N(i)} \rho(v_j) = 0$$

ainsi les systèmes 3.5 deviennent équivalents à :

$$\begin{cases} M\overrightarrow{x} = \overrightarrow{a_x} \\ M\overrightarrow{y} = \overrightarrow{a_y} \end{cases} \tag{3.6}$$

où les vecteurs $\overrightarrow{a_x}$ et $\overrightarrow{a_y}$ sont définis par

$$a_x[i] = \sum_{j \leq k, j \in N(i)} \rho(v_j)_x = \sum_{j \leq k, j \in N(i)} p_{ix}$$

$$a_y[i] = \sum_{j \leq k, j \in N(i)} \rho(v_j)_y = \sum_{j \leq k, j \in N(i)} p_{iy}$$

et M s'obtient à partir de la matrice laplacienne de G(de taille $(n-k) \times (n-k)$). Pour prouver le théorème il nous reste alors à montrer qu'une telle matrice M est inversible. Si on dénote par Q_G la matrice laplacienne de G (de taille $n \times n$), alors M est une sous-matrice de Q_G et on peut réécrire $M = Q_G[1, \ldots, k]$ [4]. Considérons maintenant le graphe G/F qu'on obtient de G avec la contraction de tous les sommets du cycle F en un seul sommet v_F : G/F possède $n+1$ sommets, et on peut supposer sans perte de généralité que v_F est le premier sommet de ce graphe. La matrice laplacienne $Q_{G/F}$ a donc taille $(n+1) \times (n+1)$, et contient M comme sous-matrice : et plus précisément $Q_{G/F}[1] = M$ (il suffit de supprimer la première ligne et colonne).

Rappelons maintenant le lemme 3.18 qui nous dit que le nombre d'arbres couvrants de G/F est donné par

$$\tau(G/F) = det(G_{G/F}[1]) = det(M)$$

Pour terminer, il suffit de remarquer que le graphe G/F possède au moins un arbre couvrant car il est connexe (étant obtenu par la contraction d'un cycle périphérique) ce qui montre que $det(M) > 0$. □

De ce lemme découle directement la méthode de Tutte pour dessiner un graphe planaire 3-connexe dans le plan. Observons d'abord que tout graphe planaire 3-connexe possède des cycles F, dits *cycles périphériques* , qui n'ont pas de *cordes* (des arêtes dans $G \setminus F$, avec les deux extrémités sur F) et tels que $G \setminus F$ est connexe : par exemple, tout cycle définissant une face d'un graphe planaire 3-connexe satisfait ces deux conditions.

La première étape consiste à choisir une application ρ_F qui envoie les sommets du cycle (de la face) F sur les sommets d'un polygone convexe $P \subset \mathbb{R}^2$ (de taille $|F|$) : bien sur, il faut s'assurer que les relations d'adjacence entre sommets dans G soient respectées par l'application ρ_F. Or, pour le lemme ci-dessus, il existe une unique extension $\rho : G \longmapsto \mathbb{R}^2$, étant de plus barycentrique. Le théorème 3.21 nous garantit alors que le dessin correspondant à la représentation ρ est bien un dessin de G dans $\mathbb{R}^2$ par des segments de droites sans croisements.

4. Rappelons qu'avec la notation $Q[1, \ldots, k]$ on indique la sous-matrice obtenue en effaçant les k premières lignes et colonnes de Q.

3.5.3 Dessiner un graphe en minimisant son énérgie

L'idée générale est très simple et basées sur le modèle physique suivant : on peut imaginer un graphe comme un ensemble de particules (ses sommets) reliés par des ressorts (ses arêtes). Si on fixe (dans le plan) les sommets d'une face (qui va devenir la face infinie), les points à l'intérieur vont se deplacer de manière à atteindre un état d'énérgie minimale : celle-ci étant l'énérgie potentielle donnée par l'ensemble de ressorts.

Le problème de dessiner un graphe dans le plan se reduit à celui de trouver une représentation qui minimise la fonction énérgie [5]

$$E := \sum_{e \in E} \texttt{longueur}(e)^2 = \sum_{(i,j) \in E} (x_i - x_j)^2 + (y_i - y_j)^2$$

Énergie d'une représentation A toute représentation ρ d'un graphe $\mathcal{G}$ ponderé (le graphe est muni d'une fonction *poids* associée aux arêtes), on peut faire correspondre une fonction *énergie*, définie comme la somme pondérée des normes au carré des distances entre points dans $\mathbb{R}^d$:

$$E(\rho) = \sum_{(i,j) \in E} w_{ij} \cdot \|\rho(v_i) - \rho(v_j)\|^2$$

Et maintenant nous avons tous les outils pour réformuler le problème de trouver un dessin d'un graphe $\mathcal{G}$ en terme du calcul d'une représentation $\rho : V(G) \longmapsto \mathbb{R}^d$ qui minimise l'énergie définie ci-dessus. Il suffit de reécrire l'énérgie de la manière suivante (où $\mathbf{p}_i := \rho(v_i)$) :

$$E(\rho) = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} \frac{1}{2} w_{ij} \cdot \|\mathbf{p}_i - \mathbf{p_j}\|^2$$

et dériver par rapport à $\mathbf{p}_i$. Comme les premiers k sommets sont fixés sur le bord de la face externe, minimiser $E(\rho)$ revient à résoudre le système de $n-k$ équations ($i = k-1 \dots n$) suivant

$$\sum_{j \in \mathcal{N}_i} w_{ij} \mathbf{p}_i = \sum_{j \in \mathcal{N}_i} w_{ij} \mathbf{p}_j$$

Exercice 3.23. *Montrer que la résolution du système donnée ci-dessus est équivalente revient à exiger que tout point interne s'exprimer comme combinaison affine de ses voisins. Justifiez qu'un tel système admet toujours une solution (Hint : réécrire le système d'équations de manière à faire intervenir la matrice laplacienne de G)*

3.5.3.1 Dessin spectral

La méthode qu'on va esquisser dans cette secion est particulièrement simple à mettre en place et fait intervenir les propriétés spectrale de la matrice laplacienne associée à un graphe (le graphe n'étant pas forcement planaire). Le procédé qui nous conduit à une représentation d'un graphe peut consiste encore une fois à résoudre un problème de minimisation d'énérgie (celle donnée par la somme ponderées des énérgies des ressorts) : la différence pricinpale est qu'ici ne fixe pas la position des sommets du bord, et on introduit d'autres contraintes.

5. Rappelon que l'énérgie potentielle d'un ressort est $\frac{1}{2}kl^2$, où l est sa longueur et k une constante

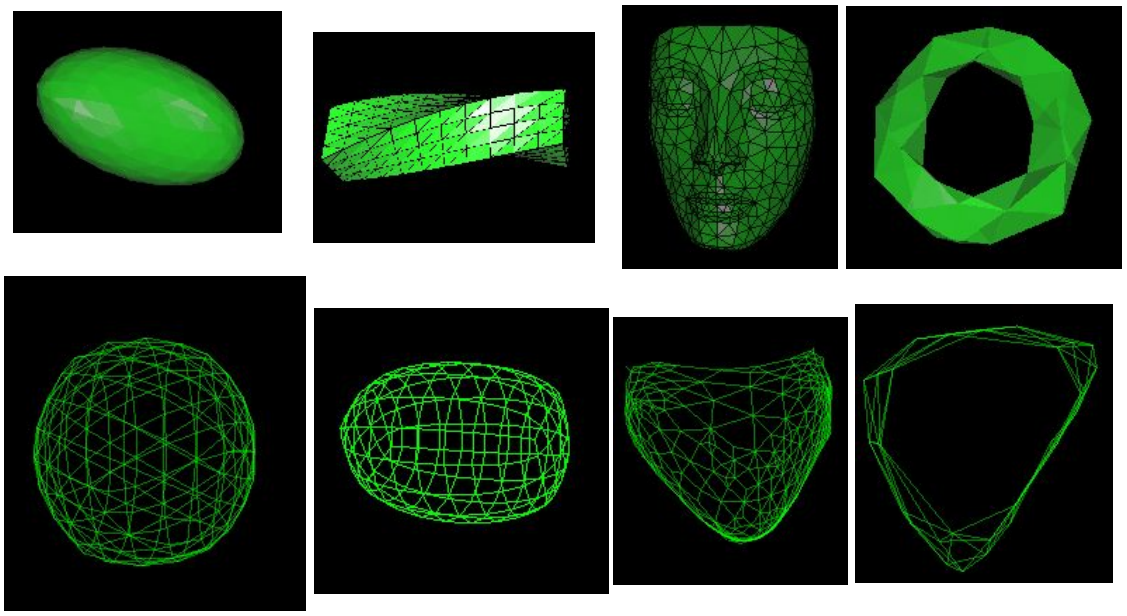


FIGURE 3.12 – *Quelques exemples de représentations de graphes en $\mathbb{R}^3$ obtenus avec la méthode spectrale. Les deux premiers graphes correspondent à des maillages homéomorphes à une sphère (il s'agit donc de triangulations planaires). Le troisième exemple (représentant Nefertiti) correspond à une triangulation avec bords. Le dernier exemple montre comment la méthode spectrale produise des dessins intéressants même lorsque le graphe n'est pas planaire (ici correspondant à un maillage de genre 1). Ces représentations ont été obtenues à l'aide des librairies* `Jcg` *et* `Jama`*.*

Idée générale Comme auparavnt on veut déterminer la représentation ρ pour laquelle l'énérgie $E(\rho)$ est minimale : on impose certaines contraintes puisque on veut un dessin de G qui ne soit pas dégéneré.

Une remarque fondamentale est que l'énérgie peut s'exprimer comme une forme quadratique qui fait intervenir la matrice laplacienne du graphe :

$$\mathbf{x}Q\mathbf{x} = \sum_{i<j} \|x_i - x_j\|^2$$

Pour éviter que tous les sommets collapsent en un seul point, on impose la contrainte suivante :

$$\mathbf{x}^T \cdot \mathbf{x} = 1$$

De plus, on vise à obtenir une représentation équilibrée, et le centre de masse va coincider donc avec l'origine, ce qui équivaut à

$$\mathbf{x}^T \cdot \mathbf{1}_n = 0$$

.

Rappelons d'abord qu'étant donné un graphe G à n sommets sa matrice laplacienne Q_G est de taille n et symétrique à valeurs réelles : on sait que ses valeurs propres $\lambda_1, \lambda_2, \ldots, \lambda_n$ sont réels, et que les vecteurs propres $v_1, \ldots, v_n$ sont orthogonaux. De plus, il est facile de voir que le vecteur $\mathbf{1}_n = [1, \ldots, 1]$ est un vecteur propre correspondant à la valeur propre $\lambda = 0$. De plus, comme la multiplicité $m(0)$ de la valeur propre 0 compte le nombre de composantes connexes, on en déduit que si le graphe G est connexe alors le $\mathbf{1}_n$ est le seul vecteur propre correspondant à la valeur propre 0.

Le théorème suivant montre que le choix d'une base de vecteurs orthogonaux conduit à une représentation de G ayant une énérgie minimale.

Théorème 3.24. *Soit $\mathcal{G}$ un graphe à n sommets, muni d'une matrice laplacienne Q, ayant valeurs propres $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ (avec $\lambda_2 > 0$). Alors l'énergie minimum d'une représentation équilibrée orthogonale de $\mathcal{G}$ dans $\mathbb{R}^d$ est donnée par*

$$\sum_{i=2}^{d+1} \lambda_i$$

Pour dessiner G il suffira de se servir des vecteurs et valeurs propres de Q_G de la manière suivante. Si on dénote par $\mathbf{p_i} = \rho(v_i) \in \mathbb{R}^d$ le point image du sommet v_i, alors on aura une jolie représentation de G en définissant :

$$\mathbf{p_i} = (p_i[1], \ldots, p_i[d]) := (\frac{v_2[i]}{\sqrt{\lambda_2}}, \frac{v_3[i]}{\sqrt{\lambda_3}}, \ldots, \frac{v_{d+1}[i]}{\sqrt{\lambda_{d+1}}}) \tag{3.7}$$

3.5.4 Dessins sur une grille et méthode de Schnyder

Un résultat majeur de la théorie des graphes des deux dernières décennies est une structure combinatoire connue sous le nom de *forêts de Schnyder*.

Les forêts de Schnyder fournissent une jolie structure combinatoire permettant de comprendre de manière fine et profonde la notion de planarité d'un graphe. Cette structure doit son appellation à W. Schnyder, qui l'introduisit sous le nom de *réaliseur*, pour les liens étroits que les forêts de Schnyder ont avec certaines propriétés des ordres partiels.

L'une des premières applications conduit justement à un nouveau critère de planarité, qui peut s'exprimer en termes de dimension d'ordres partiels(ordre d'incidence arêtes/sommets d'un graphe) [6]. Mais l'intérêt principal des forêts de Schnyder réside probablement dans leur innombrables applications conduisant à des algorithmes de dessins de graphes planaires, dont celui proposé par Schnyder constitue un premier exemple, très simple et élégant [Sch90]. Il existe plusieurs formulations possibles des forêts de Schnyder, en termes d'*étiquetage des angles*, ou de *coloriage et orientation* des arêtes, ou encore en termes d'*orientations contraintes* (où les arêtes du graphe sont orientées et le nombre d'arêtes orientées sortantes, autour de chaque sommet, est fixé). L'une des formulations plus connues est probablement celle donnée pour la famille des graphes plans maximaux, aussi connus sous le nom de *triangulations planaires*, pour lesquelles la caractérisation de Schnyder s'exprime de la manière suivante (voir Fig. 3.13 pour un exemple) :

Définition 3.25 (Schnyder [Sch90])**.** *Une **forêt de Schnyder** d'une triangulation planaire $\mathcal{T}$, ayant face externe (v_0, v_1, v_2), est une partition de ses arêtes internes en trois ensembles T_0, T_1 et T_2 d'arêtes orientées, telle que pour chaque sommet interne v les conditions suivantes sont satisfaites :*

- *v a exactement une arête sortante dans chacun des trois ensembles T_0, T_1 et T_2 ;*
- ***condition locale de Schnyder :** les arêtes incidentes à tout sommet v interne se succèdent en sens anti-horaire de la manière suivante : une arête sortante dans T_0, zéro ou plusieurs arêtes orientées entrantes dans T_2, une arête sortante dans T_1, zéro ou plusieurs arêtes entrantes dans T_0, une arête sortante dans T_2, et finalement zéro ou plusieurs arêtes entrantes dans T_1.*

6. Ce critère affirme qu'un graphe G est planaire si la dimension de l'ordre partiel d'incidence aretes/sommets est au plus 3

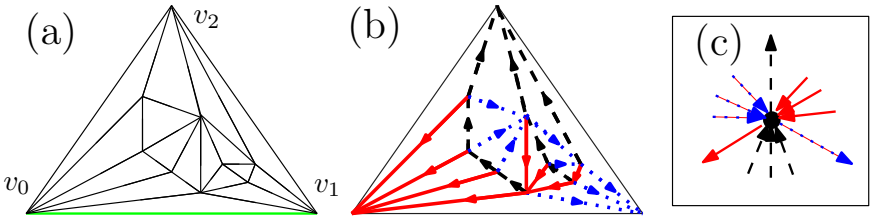


FIGURE 3.13 – *Ces images illustrent la définition et le propriétés des forêts de Schnyder d'une triangulation planaire.*

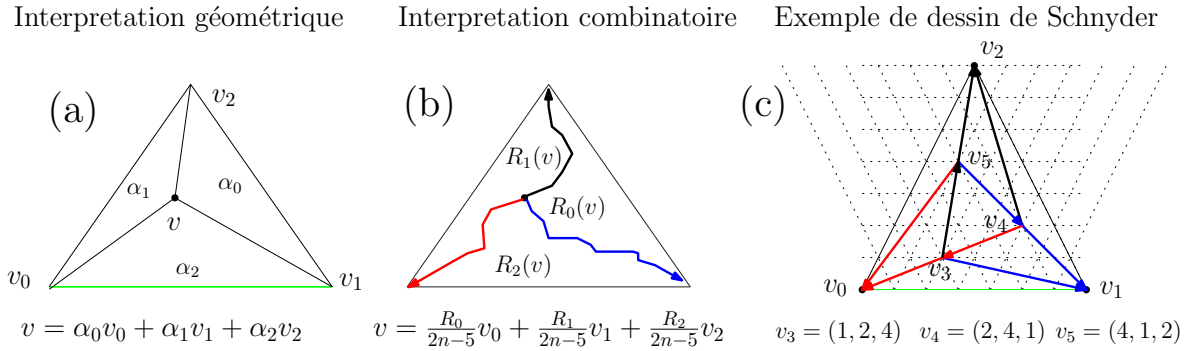


FIGURE 3.14 – *Ces images illustrent l'interprétation géométrique de la méthode de dessin de Schnyder basée sur les coordonnées barycentriques.*

Un résultat fondamental relatif aux forêts de Schnyder est que toute triangulation planaire admet une telle orientation d'arêtes. De plus, il existe un algorithme simple et de complexité linéaire permettant de calculer une forêt de Schnyder d'une triangulation donnée. Et une propriété importante des forêts de Schnyder s'exprime en termes de décompositions arborescentes, comme suit :

Lemme 3.26. *Les arêtes internes d'une triangulation planaire peuvent se partitionner en* 3 *arbres couvrants de l'ensemble des sommets, chacun ayant comme racine l'un des trois sommets incidents à la face externe. Et plus précisément on a :*

- T_0 *est un arbre couvrant de* $\mathcal{T} \setminus \{v_1, v_2\}$ *;*
- T_1 *est un arbre couvrant de* $\mathcal{T} \setminus \{v_0, v_2\}$ *;*
- T_2 *est un arbre couvrant de* $\mathcal{T} \setminus \{v_0, v_1\}$ *;*

Dessin sur une grille Nous disposons maintenant de presque tous les ingrédients pour décrire l'algorithme de dessin proposé par Schnyder.

Théorème 3.27 (Schnyder [Sch90])**.** *Toute triangulation planaire* T *à* n *sommets peut être dessiné sur une grille de taille* $(n-2) \times (n-2)$ *en temps* $O(n)$ *(le dessin étant convexe).*

Une première propriété remarquable est la suivante. Soit v un sommet interne de la triangulation. Alors il existe 3 chemins disjoints, partant de v, et ayant comme autre extrémités les 3 sommets incidents à la face externe. Pour se convaincre observons d'abord que v appartient à tous les 3 arbres (un pour chaque couleur $i = 0 \dots 2$) et qu'il existe donc 3 chemins conduisant de v aux 3 racines (qui sont les sommets de la face externe). Si, par l'absurde, ces chemins n'était pas disjoints alors il devrait exister un point u différent de v pour lequel la condition

locale de Schnyder serait violée : la condition locale décrit l'orientation et la couleur des arêtes incidentes à un sommet donné.

Un point crucial est que ces trois chemins permettent de partitionner les $2n-5$ faces internes d'une triangulation en 3 régions disjointes R_i. Cette remarque, combinée avec les propriétés des coordonnées barycentriques, permettre d'établir la validité du théorème 3.27.

La stratégie à suivre pour obtenir une représentation ρ d'un graphe dans une grille régulière est la suivante :

- sans perte de généralité, on part d'un graphe T qu'on considère triangulé ayant $n+2$ sommets ;
- on calcule une forêt de Schnyder de T (en temps $O(n)$) ;
- on choisit 3 points p_0, p_1, p_2 qui seront les images des sommets v_0, v_1, v_2 ;
- pour tout sommet interne v, on calcule le nombre f_i de triangles contenus dans la région R_i
- on calcule les trois coordonnées barycentriques suivantes pour tout sommet interne v :

$$\alpha_v = \frac{f_0^v}{2n-5},\ \beta_v = \frac{f_1^v}{2n-5},\ \gamma_v = \frac{f_2^v}{2n-5}$$

- l'application ρ est définie comme suit (v sommet interne) :

$$\rho(v) = \alpha_v p_0 + \beta_v p_1 + \gamma_v p_2$$

Interprétation combinatoire d'une propriété géométrique Avant de fournir une preuve plus détaillée, il est convenable de faire quelques remarques. Comme déjà observé (Chapitre 1) un point v interne à un triangle (p_0, p_1, p_2) peut s'exprimer comme combinaison barycentriques des extrémités p_0, p_1 et p_2 : dans ces cas les coefficients représentent les aires (normalisées), des trois triangles définis par v et les trois segments du triangle.

Or, dans le contexte combinatoire un fait une chose similaire : on veut exprimer un point $\rho(v)$ comme combinaison barycentriques des points images des sommets de la face externe. La seule différence est qu'ici "les aires" sont représentées par le nombre de faces contenues dans chacune des trois régions R_0, R_1 et R_2.

Pour valider ce procédé il ne reste qu'à montrer que l'application ρ est une application barycentrique : dans ce cas il en résulte que le dessin de la triangulation T est sans croisements.

3.5.5 Preuve du Théorème barycentrique de Tutte (Lemme 3.21)

Nous allons prouver le théorème barycentrique de Tutte dans un cas spécial. En particulier, on va montrer qu'une représentation barycentrique $\rho : V(G) \to \mathbb{R}^2$ définit un dessin planaire d'un graphe (sans croisements), avec les propriétés supplémentaires suivantes (qu'ici on considère vérifiées) :

- à l'exception de la face correspondant au cycle F (la face externe), toutes les faces sont des triangles.
- les images des triangles de G, par l'application ρ, sont non dégénérés (leurs intérieurs sont non vides).
- $\rho(v)$ appartient à l'intérieur de l'enveloppe convexe de ses voisins.
- l'image $\rho(v)$ de tout sommet interne v est un point appartenant à l'intérieur du polygone $P = \rho(F)$.

La première propriété décrit la locale planarité de la représentation ρ comme exprimé ci-dessous (la preuve suit la présentation fournie par Eric Colin de Verdière [dV03]) :

Lemme 3.28. *Avec les hypothèses ci-dessus, les images de deux triangles de G, partageant un sommet commun, ne se superposent pas (leurs intérieurs sont disjoints).*

Considérons un dessin planaire de G, et dénotons par $\alpha(v)$ la somme des angles des triangles autour de v. Il est facile de voir que $\alpha(v_i) = 2\pi$ pour tout sommet interne de G, alors que pour tout sommet v_F de la face externe $\alpha(v_F)$ est égal à l'un des angles du polygone convexe P image de la face F. Étant donné un sommet v appartenant à une face (u, v, w) de G, soit $\theta(u, v, w)$ l'angle définit par les arêtes $(\rho(u), \rho(v))$ et $(\rho(w), \rho(v))$. Dénotons maintenant par $\sigma(v)$ la sommes des angles des triangles qui sont images des faces de G par ρ.

Comme il est facile de deviner, dans la partie restante de la preuve on va prouver que $\sigma(v) = \alpha(v)$ pour tout sommet v.

Claim 3.29. *Pour tout sommet v, $\sigma(v) \geq \alpha(v)$. Et $\sigma(v) = \alpha(v)$ si et seulement si les triangles ne se superposent pas.*

Démonstration. Considérons un sommet interne v, avec d voisins $v_1, \ldots, v_d$: pour les hypothèses ci-dessus ce sommet appartient à l'intérieur de l'enveloppe convexe de ses voisins et en particulier, il est contenu dans un triangle qui est l'enveloppe convexe de trois voisins (il suffit de trianguler le polygone convexe pour en se rendre compte). Il existe alors trois sommets v_i, v_j et v_k (avec $1 \leq i < j < k \leq d$) tels que les 3 angles $\theta(v_i v v_j)$, $\theta(v_j, v, v_k)$ et $\theta(v_k, v, v_i)$ sont égaux à 2π.

Mais en considérant la contribution des angles des autres triangles (correspondants aux voisins de v), on obtient

$$\sum_{q=i}^{j-1} \theta(v_q v v_{q+1}) \geq \theta(v_i v v_j)$$

et les deux autres relations similaires : $\sum_{q=j}^{k-1} \theta(v_q v v_{q+1}) \geq \theta(v_j v v_k)$ et $\sum_{q=k}^{i-1} \theta(v_q v v_{q+1}) \geq \theta(v_k v v_i)$ (les indices sont pris modulo d). D'où la relation :

$$\sigma(v) = \sum_q \theta(v_q v v_{q+1}) \geq \theta(v_i v v_j) + \theta(v_j v v_k) + \theta(v_k v v_i) \geq 2\pi$$

□

Claim 3.30. *Pour tout sommet interne v on a $\sigma(v) = \alpha(v)$*

Démonstration. En rappelant que la somme des angles internes d'un polygone P ayant $|P|$ sommets est $(|P|-2)\pi$ (et que la somme des angles d'un triangle est π), si on dénote par $|V \setminus F|$ l'ensemble des sommets internes de G et f le nombre de ses faces triangulaires on a :

$$\sum_{v \in V} \alpha(v) = \sum_{v \in V \setminus F} \alpha(v) + \sum_{v \in F} \alpha(v) = 2\pi |V \setminus F| + (|F| - 2)\pi \leq \sum_{v \in V} \sigma(v) = \pi f \tag{3.8}$$

On va maintenant réécrire la formule d'Euler pour le graphe G, qui a toutes ses faces triangulaires, à l'exception de la face externe qui a taille $|F|$ (e dénote le nombre d'arêtes internes de G) :

$$n - (e + |F|) + f \; = \; (|V \setminus F| + |F|) - (e + |F|) + (t + 1) = 2 \tag{3.9}$$

En sachant que toute arêtes interne appartient exactement à deux triangles, et que les arêtes externes n'apparaissent qu'une fois dans un triangle, on peut compte le nombre d'arêtes comme suit :

$$3t = 2e + |F| \tag{3.10}$$

Et maintenant les équations 3.9 et 3.10 suffisent pour montrer que $2\pi|V \setminus F| + (|F|-2)\pi = \pi f$, ce qui permet d'obtenir $\sum_{v \in V} \alpha(v) = \sum_{v \in V} \sigma(v)$, à partir de la relation 3.8. Et pour terminer, le claim précédent nous que $\sigma(v) \geq \alpha(v)$, ce qui implique $\sigma(v) = \alpha(v)$ pour tout sommet v. □

Pour déduire la validité du Lemme 3.28 il suffit de combiner les deux claims : pour tout sommet interne la somme des angles des triangles incidents est exactement 2π, ce qui montre que la représentation ρ est localement planaire.

3.5.6 Notes bibliographiques

Il existe de nombreux textes traitant les différents aspects de la théorie des graphes. Le lecteur qui serait intéressé à approfondir les propriétés algébriques des graphes, pourrait apprécier la lecture du livre de Godsil [GR01] qui traite de manière pédagogique et exhaustive ce sujet. Notamment, nous nous sommes partiellement inspirés du Chapitre 13 de [GR01] pour la présentation des propriétés spectrales de la matrice laplacienne d'un graphe.

La présentation de la preuve du théorème de Koebe suit celle fournie par les notes de cours de V. Koltun, qui décrivent aussi de manière très détaillé d'autres avancées récentes de la géométrie algorithmique.

Il existe plusieurs manières de montrer la validité de la méthode barycentrique de Tutte : en particulier, plusieurs preuves du lemme 3.21 ont été proposées. Celle que nous avons présentée ici a le mérite d'être simple et concise, étant due à Eric Colin de Verdière (une présentation détaillée est fournie dans sa thèse de doctorat).

3.6 Graphes et petits séparateurs

Intuitivement un graphe est*séparable* (ou avec *petit séparateurs*) si tous ses sous-graphes peuvent être partitionnés en deux parties ayant environ la même taille, juste en supprimant un nombre relativement limité de sommets (il existe plusieurs notions de séparabilité, en terme de sommets ou d'arêtes).

Plus précisément, considérons une classe $\mathcal{C}$ de graphes avec la propriété que tout sous-graphe d'un graphe $G \in \mathcal{C}$ appartient encore à $\mathcal{C}$.

Définition 3.31. *Un graphe $G \in \mathcal{S}$ ayant n sommets est* séparable *s'il existe un ensemble C de $f(n)$ sommets (appelé* coupe *ou $f(n)$-*séparateur*) dont la suppression déconnecte G en deux sous-graphes A et B (appartenant à la classe C) ayant chacun au plus αn sommets, où $\alpha < 1$, et $f(n) = o(n)$.*

Dans cette définition on voit intervenir la notion de séparabilité par sommets, mais il est aussi possible d'en définir une autre, légèrement différente, en termes d'arêtes. Les deux notions sont différentes : étant donnée un séparateur ayant un petit nombre de sommets on peut trouver une petite coupe d'arêtes, mais l'opposé est faux en général.

De manière générale on considère des graphes ayant des séparateurs de taille $f(n) = O(n^c)$, avec $c < 1$. Dans ce cadre, ils s'avère que certaines classes de graphes sont spécialement intéressantes : par exemple, les graphes planaires admettent des séparateurs de taille $O(n^{1/2})$ [LT79], et plus généralement les maillages surfaciques de genre borné satisfont les conditions de la Définition 3.31. Ou encore, certains maillages volumiques *bien formés* en $\mathbb{R}^d$ ont des séparateurs de taille $O(n^{1-\frac{1}{d}})$ [MTTV97].

On peut commencer par considérer quelques cas particuliers de graphes planaires, pour lesquels on peut facilement trouver un petit séparateur.

Exercice 3.32. *La classe des arbres plans admet un 1-séparateur.*

Avant d'introduire la construction de Liption/Tarjan, il est facile de voir que des petits séparateurs existent aussi pour les maillages réguliers (tous les sommets ont le même degré) :

Exercice 3.33. *Toute grille régulière G de taille $\sqrt{n} \times \sqrt{n}$ admet un séparateur de taille $\sqrt{n}$.*

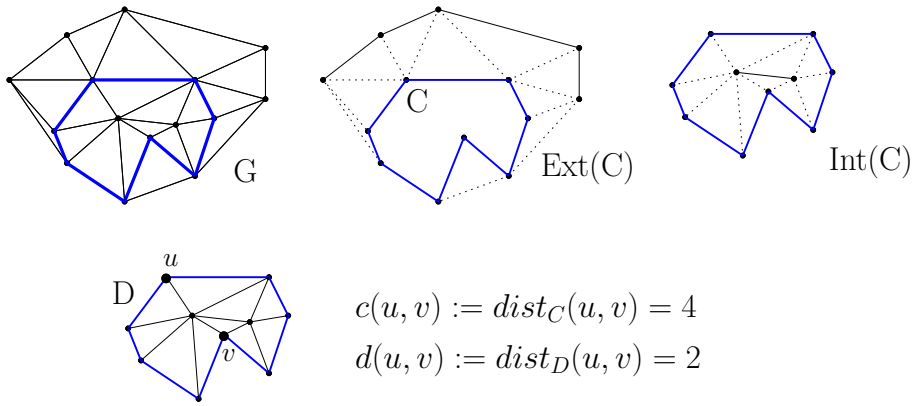


FIGURE 3.15 – *Ces images illustrent la preuve du théorème de Lipton et Tarjan pour le calcul d'un petit séparateur d'un graphe planaire.*

3.6.1 Séparateurs et graphes planaires

Le but de cette section est de fournir une preuve d'un résultat fondamental établi pour la première fois par R. Lipton et R. Tarjan [LT79].

Théorème 3.34 (Lipton et Tarjan)**.** *Soit $\mathcal{G}$ un graphe planaire ayant n sommets. Alors il existe un $\frac{2}{3}$-séparateur contenant au plus $\sqrt{8n}$ sommets.*

Pour la preuve, nous suivons le raisonnement proposé par Alon, Seymour et Thomas [AST94] [7].

Tout d'abord fixons quelques notations et hypothèses. Premièrement, on peut supposer sans perte de généralité que le graphe $\mathcal{T}$ est une triangulation simple (sans boucle n'y d'arête multiple). Rappelons qu'on peut toujours considérer un plongement planaire de G avec $n \geq 3$ sommets et trianguler toutes ses faces polygonales : tout séparateur de cette triangulation (ayant le même nombre de sommets) sera aussi un séparateur de G satisfaisant les conditions du théorème 3.34.

Considérons un cycle simple C : par construction (et pour le théorème de la courbe de Jordan) les sommets de $G \setminus C$ sont séparés en deux sous-ensembles non adjacents, $Int(C)$ et $Ext(C)$ respectivement. On fixe un paramètre $k = \lfloor\sqrt{2n}\rfloor$, et on va considérer, parmi tous les cycles simples C, ceux qui satisfont les conditions suivantes :

— $|C| \leq 2k = \lfloor 2\sqrt{2n}\rfloor$,
— $|Ext(C)| < \frac{2n}{3}$,

En plus, on va demander que la condition de minimalité suivante soit vérifiée :

— la quantité $|Int(C)| - |Ext(C)|$ est plus petite que possible.

On peut tout de suite observer qu'un cycle satisfaisant les deux premières conditions existe toujours : en effet le bord de la face externe est bien un cycle de taille 3, satisfaisant clairement la deuxième condition. Soit D le sous-graphe de G défini par l'intérieur de C auquel on ajoute le cycle C. Pour tout couple de sommets $u, v \in C$ on dénote par $dist_D(u,v)$ et $dist_C(u,v)$ les distances entre les deux sommets dans le graphe D et dans le cycle C respectivement : la distance étant définie comme la taille du plus court chemin entre deux sommets (pour un exemple, voir la Fig. 3.15). Pour prouver le théorème 3.34 on procède en raisonnant par l'absurde : on va supposer que $|Int(C) \geq \frac{2n}{3}|$ et on fait appel aux deux lemmes décrits ci-dessous.

Lemme 3.35. *Soit C un cycle simple satisfaisant les conditions ci-dessus. Si $|Int(C)| \geq \frac{2n}{3}$,*

7. La stratégie de preuve proposée ici suit la présentation fournie par Jeff Erickson.

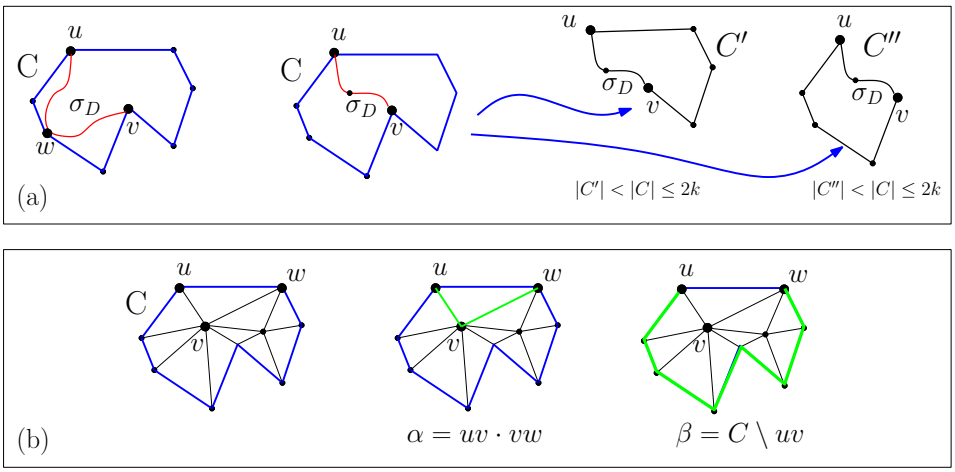


Figure 3.16 – *Ces figures illustrent les deux lemmes utilisés dans la preuve du théorème de Lipton et Tarjan.*

alors pour tout couple de sommets u et v appartenant au cycle C on a

$$d(u,v) = c(u,v)$$

Démonstration. Raisonnons par l'absurde, en supposant qu'il existe deux sommets distincts u et v tels que $d(u,v) < c(u,v)$. Observons d'abord que $d(u,v) \leq c(u,v)$, tout simplement parce que C est un sous-graphe de D.

Maintenant on va choisir, parmi tous les couples de sommets u, v pour lesquels $d(u,v) < c(u,v)$, ceux qui minimise la distance $d(u,v)$. Dénotons par σ_D le plus court chemin, dans D, allant de u à v. On va d'abord montrer que le chemin σ_D ne contient pas de sommet $w \in C$ $(w \neq u, v)$ (voir Fig. 3.16(a)). En effet, s'il y avait un tel sommet w, alors on pourrait écrire $d(u,w)+d(w,v) = d(u,v) < c(u,v) \leq c(u,w)+c(w,v)$, ce qui conduirait à prouver qu'au moins l'une des inégalités $d(u,w) < c(w,v)$, ou bien $d(w,v) < c(w,v)$, devrait être vérifiée. Mais si on avait, par exemple, $d(u,w) < c(w,v)$ alors on aurait trouvé un autre couple u, w tel que $d(u,w) < d(u,v)$, ce qui contredit la minimalité du couple u, v.

Puisque on vient de prouver que le cycle σ_D ne contient pas de sommet $w \in C$, on peut alors définir deux nouveaux cycles simples C' et C'', obtenus en coupant le cycle C en u et v, et en faisant la concaténation avec σ_D (comme illustré par les images de droite de la Fig. 3.16(a)). Puisque on a supposé par absurde que $d(u,v) < c(u,v)$, alors on sait que $|\sigma_D| < c(u,v)$, et donc $|C'| < |C|$ et $|C''| < |C|$. En supposant, sans perte de généralité, que $|Int(C')| \geq |Int(C'')|$, on arrive à établir les relations suivantes :

$$n - |Ext(C')| = |Int(C')| + |V(C')| > \frac{1}{2}(|Int(C')| + |Int(C'')| + |V(\sigma_D)| - 2) = \frac{1}{2}|Int(C)| \geq \frac{n}{3}$$

et donc le cycle C' satisfait les deux condition définies auparavant, car $|Ext(C')| < n - \frac{n}{3} = \frac{2n}{3}$, et $|C'| < |C| \leq 2k$. Mais alors on atteint une contradiction : puisque $Ext(C') > Ext(C)$ et $Int(C') < Int(C)$ on obtient

$$|Int(C')| - |Ext(C')| < |Int(C)| - |Ext(C)|$$

alors qu'on avait supposé par absurde que la quantité $|Int(C)| - |Ext(C)|$ était minimisée par le cycle C. □

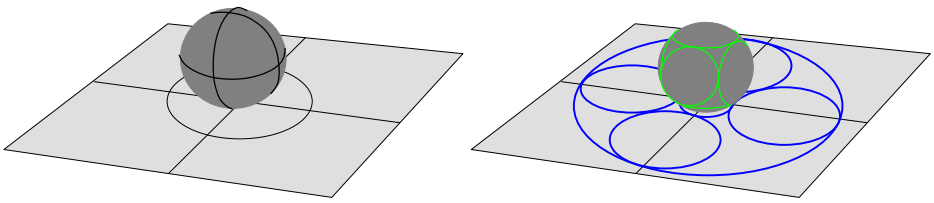


FIGURE 3.17 – *Tout graphe planaire est isomorphe au graphe d'intersection de calottes circulaires sur la sphère : sur la gauche un graphe planaire dessiné sur la sphère, dont la représentation de Koebe est présentée sur la droite.*

Lemme 3.36. *Soit C un cycle simple satisfaisant les conditions du lemme précédent. Alors C contient exactement $2k = 2\lfloor\sqrt{2n}\rfloor$ sommets.*

Démonstration. Encore une fois on procède par l'absurde, en supposant que la taille du cycle est $|C| < 2k$. Considérons une arete (u, w) du cycle C : comme G est triangulé l'arete (u, w) est incidente à deux faces triangulaires, l'une d'entre elles, dénotée (u, v, w), se trouvant dans D. On peut maintenant définir deux distincts $\alpha = uv \cdot vw$ et $\beta = C \setminus (u, w)$: α et β sont disjoints car $Int(C) \neq \emptyset$, puisque on est en train de supposer que $|Int(C)| \geq \frac{2n}{3}$. Il en résulte alors, grace au lemme précédent, que le sommet v ne peut pas appartenir à C. On peut construire maintenant un nouveau cycle simple $\overline{C} = \alpha \cdot \beta$ obtenu par la concaténation des deux chemins α et β (voir la Figure 3.16(b)). Clairement $|\overline{C}| \leq 2k$, car $|\overline{C}| = |C| + 1$ et $|C| < 2k$. De plus, on a que $|Ext(\overline{C})| < \frac{2n}{3}$. Mais alors on vient de construire un cycle $\overline{C}$ qui satisfait les deux première conditions, et pour lequel la quantité $|Int(\overline{C})| - |Ext(\overline{C})|$ est plus petite que $|Int(C)| - |Ext(C)|$, ce qui viole la minimalité du cycle C, d'où la contradiction : observons que $|Int(\overline{C})| < |Int(C)|$ puisque par construction $\overline{C}$ contient tous les sommets de C, plus le sommet v. □

On peut finalement esquisser la preuve du théorème de Lipton et Tarjan :

Preuve du théorème 3.20. Considérons un cycle simple $C = \{v_0, v_1, \ldots, v_{2k-1}, v_{2k} = v_0\}$ de taille $2k$. Pour le premier lemme ci-dessous on sait que $d(v_0, v_k) = c(v_0, v_k) = k$, ainsi le plus court chemin dans D allant de v_0 à v_k a taille $|\sigma_D(v_0, v_k)| = k$.

Alors, pour le théorème de Menger, il existe $k + 1$ chemins disjoints, notés $\pi_0, \ldots, \pi_k$ allant des sommets $\{v_0, \ldots, v_k\}$ aux sommets $\{v_k, \ldots, v_{2k}\}$: où le i-ième chemin est $\pi_i : v_i \longrightarrow v_{2k-i}$. Or, la taille des $k + 1$ chemins π_i est clairement inférieure au nombre total de sommets de G : $n \geq \sum_{i=0}^{k} |V(\pi_i)|$. En évaluant la taille totale de tous ces chemins, on obtient donc :

$$n \geq \sum_{i=0}^{k} |V(\pi_i)| \geq \sum_{i=0}^{k} \min\{2i + 1, 2k - 2i - 1\} \geq \frac{(k+1)^2}{2}$$

d'où $k \leq \sqrt{2n} - 1$, ce qui conduit à une contradiction, puisque on avait fixé $k = \lfloor\sqrt{2n}\rfloor$. □

3.6.2 Séparateurs géométriques

Il existe d'autres intéressantes caractérisations des séparateurs planaires. En particulier, un théorème de séparation pour les graphes planaires peut-être trouvé à l'aide de la caractérisation fournie par le théorème de Koebe (voir théorème 3.9), comme suggéré par Spielman et Teng [ST96]. Tout d'abord on va fournir une reformulation du théorème de Koebe (illustrée par la Figure 3.17) :

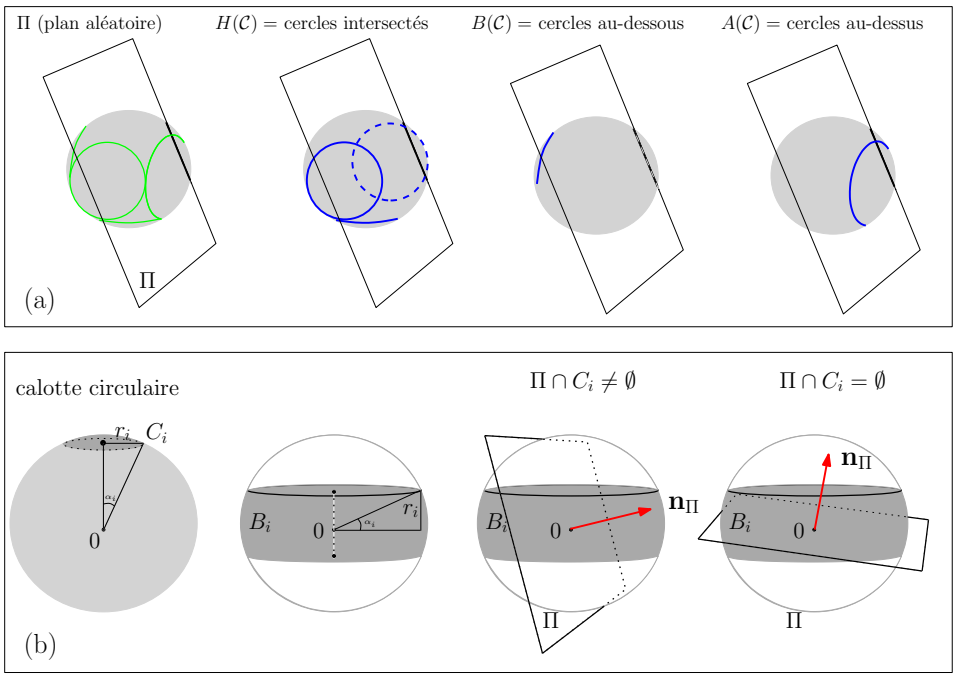


Figure 3.18 – *Ces images illustrent l'application du théorème de Koebe au cas des séparateurs géométriques.*

Théorème 3.37. *Tout graphe planaire à n sommets est isomorphe au graphe d'intersection de n calottes circulaires $\{C_1, \ldots, C_n\}$ sur la sphère (leurs intérieurs étant disjoints).*

Considérons la collection de cercles $\mathcal{C} = \{C_1, \ldots, C_n\{$ qui définissent les n calottes circulaires correspondantes aux n sommets du graphe G. Etant donné un plan $\Pi \subset \mathbb{R}^3$ passant par l'origine, on peut distinguer les cercles de $\mathcal{C}$ en trois catégories. On va dénoter par $H(\mathcal{C})$ les cercles qui sont intersectés par Π, alors que $B(\mathcal{C})$ et $A(\mathcal{C})$ dénote les cercles qui se trouve au-dessous et au-dessus de Π respectivement (comme illustré par la Figure 3.18(a)).

Lemme 3.38. *Soit $\{C_1, \ldots, C_n\}$ une collection de n calottes sur la sphère unité (dont les intérieurs sont deux à deux disjoints). Etant donné un plan aléatoire Π passant par l'origine, l'espérance du nombre de calottes intersectés par Π est au plus $2\sqrt{n}$.*

Démonstration. La première étape consiste à évaluer le nombre de plans qui intersèctent une calotte donnée sur la sphère. Observons qu'une calotte C_i est définie par l'intersection de la sphère avec un plan donné : leur intersection est un cercle de rayon r_i. Etant donné le plan aléatoire Π on peut définir le vecteur normal n qui est orthogonal à Π et passe par l'origine. On peut facilement exprimer le fait que le plan Π intersècte ou pas une calotte C_i en termes des positions du vecteur n.

Il suffit de remarquer que $\Pi \cap C_i \neq \emptyset$ si et seulement si $n_i \in B_i$, où B_i est la portion de la sphère qu'on obtient en coupant la sphère avec deux hyperplans parallèles, et dont la distance à l'origine est r_i (comme illustré par la Fig. 3.18(b)). Ainsi, la probabilité que Π intersècte une calotte C_i est égale à la probabilité qu'un point choisi aléatoirement sur la sphère unité S^2 appartiennent à B_i, ce qui est proportionnel à l'aire de B_i, comme exprimé par

$$Prob(\Pi \cap C_i \neq \emptyset) \;=\; Prob(\mathbf{n} \in B_i) = \frac{aire(B_i)}{aire(S^2)} \;=\; \frac{4\pi r_i}{4\pi}$$

De manière similaire, l'espérance du nombre de calottes intersectées par Π est égale à l'espérance du nombre des B_i contenant un point aléatoire n_Π :

$$E(|H(\mathcal{C})|) = E(|\{B_i \text{ intersectés par un point } \mathbf{n} \text{ aléatoire}\}|) = \sum_i r_i$$

Pour évaluer la quantité $\sum_i r_i$ on va procéder comme suit. Observons que l'aire d'un calotte C_i de rayon r_i est au moins πr_i^2, alors que l'aire de la sphère est 4π. Puisque les calottes C_i forment un disk-packing sur la sphère, on a $\sum_i aire(C_i) < 4\pi$, d'où on obtient donc $\sum_i r_i^2 < 4$. Ce qui conduit à la relation suivante :

$$\frac{1}{n}(\sum_i^n r_i)^2 = n(\frac{1}{n}\sum_i^n r_i)^2 \leq n(\frac{1}{n}\sum_i^n r_i^2) = \sum_i^n r_i^2 < 4$$

permettant d'établir $E(|H(\mathcal{C})|) = \sum_i r_i < 2\sqrt{n}$. □

Nous pouvons maintenant établir un théorème de séparation pour les graphes planaires.

Théorème 3.39 (Spielman et Teng)**.** *Tout graphe planaire G ayant n sommets admet un $\frac{3}{4}$-séparateur de taille au plus $2\sqrt{n}$.*

Démonstration. Comme il est désormais facile à deviner, l'existence d'un séparateur géométrique fait appel au théorème de Koebe, et peut se montrer comme suit.

Etant donné un graphe planaire G, on considère une représentation de Koebe $\{D_1, \ldots, D_n\}$ de G, où les n sommets correspondent aux points $\{p_1, \ldots, p_n\}$ (centres des disques D_i). Comme suggéré par le Théorème 3.37, on peut utiliser une application conforme $\Phi : \mathbb{R}^2 \longrightarrow S^2$ pour faire correspondre aux disques D_i des calottes circulaires C_i dont les intérieurs son deux à deux disjoints. De plus, on va demander que l'origine soit un center point de l'ensemble $\{\phi(p_1), \ldots, \Phi(p_n)\} \subset S^2$: cela est toujours possible comme montré par Miller et al. (pour la définition d'un center point, voir le Chap. 1). Ainsi, il suffit d'appliquer le lemme précédent à la collection de calottes $\{\Phi(D_1), \ldots, \Phi(D_n)\}$ (ces calottes sont bien définies, puisque l'application conforme Φ préserve les cercles). Comme déjà montré, un plan aléatoire Π passant par l'origine sépare les calottes $C_i = \Phi(D_i)$ en trois collections $A(\mathcal{C})$, $B(\mathcal{C})$ et $H(\mathcal{C})$, de manière à ce que le nombre de calottes intersectés par Π, donné par $|H(\mathcal{C})|$, est au plus $2\sqrt{n}$. Pour terminer la preuve du théorème il reste à montrer que les tailles des ensembles $A(\mathcal{C})$ et $B(\mathcal{C})$ sont au plus $\frac{3n}{4}$. Pour cela on exploite le fait que l'origine est un center point : on sait alors que tout plan passant par l'origine partitionne les points $\{\Phi(p_1), \ldots, \Phi(p_n)\}$ en deux sous-ensembles de taille au plus $\lfloor \frac{3n}{4} \rfloor$.

En conclusion, on a trouvé un sous-ensemble de taille au plus $2\sqrt{n}$ des points $\{p_1, \ldots, p_n\}$ qui constitue un séparateur du graphe G : ces sommets correspondent aux centres des calottes dans $H(\mathcal{C})$. De plus, les sommets restant de G sont partitionnés en deux sous-ensembles, de taille au plus $\lfloor \frac{3n}{4} \rfloor$: ils correspondent aux calottes dans $A(\mathcal{C})$ et $B(\mathcal{C})$ respectivement. Ce qui conclut la preuve. □

3.7 Applications algorithmiques des séparateurs

3.7.1 Localisation planaire

Les propriétés des petits séparateurs permettent d'attaquer de manière naturelle certaines classes de problèmes pouvant tirer partie d'une décomposition récursive des graphes. Un exemple est donnée par la structure conçue par Edelsbrunner, Guibas et Stolfi [EGS86] permettant de résoudre le problème de la localisation planaire.

Théorème 3.40. *Soit $\mathcal{T}$ une triangulation planaire à n sommets. Alors il existe une structure de données de taille $O(n \log n)$ permettant de répondre aux requêtes en temps $O(\log n)$.*

3.7.2 Compression et codage de graphes

Les petits séparateurs ont trouvé récemment des applications dans le domaine de la compression et codage de structures géométriques : en effet, pour certaines classes d'objets admettant des petits séparateurs (tels que les graphes planaires, les maillages surfaciques et certains maillages volumiques) il est possible de concevoir une stratégie de partitionnement qui permet de représenter de manière assez compacte l'information de connectivité.

Rappelons d'abord que les représentations usuelles d'un maillage surfacique de taille n nécessitent en général $O(n \log n)$ bits : c'est le cas, pas exemple, des représentations explicites basées sur des pointeurs (telles que Half-edge ou Quad-edge), ou d'autres formes de stockage sur disque (tel que les fichiers au format OFF). Cela dépend du fait que dans un maillage de taille n il y a $O(n)$ relations d'incidences (entre sommets, faces et arêtes), dont tenir compte : et puisque il y a $O(n)$ sommets, chaque référence coûte $O(\log n)$ bits.

Pour ce faire une idée, on peut considérer l'exemple des arbres binaires de taille n. Puisque chacun des n sommets a deux enfants, une représentation explicite par pointeurs d'un tel arbre nécessite environ $2n \log n$ bits. Mais des arguments de comptage nous disent que le nombre des arbres binaires à n sommets est donné par $C_n = \frac{1}{n+1}\binom{2n}{n}$: ainsi, pour coder un tel arbre, $\log_2 C_n \approx 2n + o(n)$ bits devraient suffire.

Dans le cas des graphes planaires (et plus généralement des graphes plongés sur des surfaces de genre borné), des résultats de comptage similaires existent, et permettent en effet de montrer que pour coder un graphe planaire on a besoin d'un nombre linéaire de bits. Même sans avoir recours à des techniques d'énumération (plutôt sophistiquées d'ailleurs), on peut commencer à coder des graphes (et donc énumérer) à l'aide du résultat suivant.

Exercice 3.41. *Soit G un graphe planaire ayant n sommets. Montrer que G peut être codé avec au plus $O(n)$ bits.* ***Hint*** *Une stratégie (assez générale) à suivre consiste à calculer un arbre couvrant B des sommets du graphe G. Le codage de G revient donc à coder l'arbre B et l'ensemble des aretes dans $G \setminus B$. Rappelons qu'un arbre de taille n peut se coder efficacement à l'aide d'un mots de parenthèses équilibrées de taille $2n$.*

Une stratégie différente (ne faisant pas intervenir d'arbres couvrants), consiste à calculer une décomposition récursive d'un graphe G à l'aide des séparateurs. Cet approche est assez générale, permettant ainsi d'élargir la classe des graphes qu'on peut coder avec un nombre linéaire de bits : elle s'applique à tous les graphes admettant des petits séparateurs.

Théorème 3.42. *Soit G un graphe à n sommets appartenant à une classe qui admet des séparateurs par arêtes de taille $O(n^{1-\varepsilon})$ (pour $\varepsilon > 0$). Alors G peut être représenté avec $O(n)$ bits.*

Esquisse de la preuve. L'idée est à la fois simple et très générale. On suppose de vouloir représenter un graphe à l'aide des listes d'adjacences : pour tout sommet du graphe, en plus de son degré, on stoque aussi la liste des indices de ses voisins. De plus, pour réduire la taille des indices, on adopte un codage par différences : si les voisins d'un sommet u de degré d ont indices $\{u_0, u_1, \ldots, u_{d-1}\} \subset \{0, \ldots, n-1\}$, alors on va représenter les voisins de u par les d entiers $\{u_0, (u_1 - u_0), (u_2 - u_1), \ldots, (u_{d-1} - u_{d-2})\}$. Or, la mise en place de cette méthode de manière naive conduit à encore à des représentations de taille $O(n \log n)$. La stratégie qu'on va esquisser consiste à renuméroter les sommets du graphe de manière à ce que deux sommets proches (dans le graphe) aient des indices assez proches.

On commence par un construire un *arbre de séparation* B où à chaque noeud on fait correspondre un sous-graphe de G. L'arbre B s'obtient en partitionnant le graphe G récursivement à l'aide des séparateurs. On part du graphe G, qui correspond à la racine de B, et on calcule une partition $G = G_1 \cup G_2$. Ainsi la racine de G aura deux sous-arbres, dont les racines correspondent aux sous-graphes G_1 et G_2. On répète récursivement ce procédé tant que tous les sommets ont été séparés : les feuilles de l'arbre B correspondent ainsi aux sommets du graphe original G. Et finalement on va numéroter les sommets de G, chacun apparaissant dans une feuille de G : cela peut se faire en réalisant un parcours en profondeur des noeuds de B.

Pour conclure, il reste à montrer qu'avec une telle numérotation, le cout pour le stoquage des indices des sommets (plus précisément leurs différences) se réduit à $O(n)$ bits. Pour évaluer la taille de la représentation d'un graphe à n sommets, dénotée par $C(n)$ on raisonne de manière récursive. Pour représenter G il faut disposer de : la représentation de G_1 (de taille $C(G_1)$), la représentation de G_2 (de taille $C(G_2)$), ainsi de l'information nécessaire à recoller G_1 et G_2 pour obtenir G. Or, la manière dont on a découpé G en deux dépend du séparateur S qu'on a calculé : et puisque par hypothèse la taille du séparateur est $|S| = O(n^{1-\varepsilon})$, l'information nécessaire pour recoller est $C(S) = O(|S| \log |S|)$. Il ne reste alors qu'à exprimer une récurrence qui permet d'exprimer $C(n)$, en termes de $C(G_1)$, $C(G_2)$ et de $C(S)$. Et pour conclure, il suffit d'observer qu'une telle récurrence possède une solution $C(n)$ telle que $C(n) = O(n)$. □

Bibliographie

[AST94] Noga Alon, Paul D. Seymour, and Robin Thomas. Planar separators. *SIAM J. Discrete Math.*, 7(2) :184–193, 1994.

[dV03] Eric Colin de Verdière. *Raccourcissement de courbes et décomposition de surfaces.* PhD thesis, Université Paris 7, 2003.

[EGS86] Herbert Edelsbrunner, Leonidas J. Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2) :317–340, 1986.

[GR01] C. Godsil and G. Royle. *Algebraic Graph Theory*, chapter "The Laplacian of a graph", pages 279–306. Springer-Verlag, 2001.

[LT79] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36 :2 :177–189, 1979.

[MTTV97] Gary L. Miller, Shang-Hua Teng, William P. Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1) :1–29, 1997.

[Sch90] Walter Schnyder. Embedding planar graphs on the grid. In *Proc. of the First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 138–148, 1990.

[ST96] Daniel A. Spielman and Shang-Hua Teng. Disk packings and planar separators. In *Symposium on Computational Geometry*, pages 349–358, 1996.

[Tut60] W. Tutte. Convex représentations of graphs. *Proc. Lond. Math. Soc.*, 10 :304–320, 1960.

[Tut63] W. Tutte. How to draw a graph. *Proc. Lond. Math. Soc.*, 13 :743–768, 1963.

Deuxième partie

Les développements plus récents

Chapitre 4

Reconstruction de formes géométriques

Le problème de la reconstruction est le suivant : étant donné un nuage de points dans $\mathbb{R}^d$, supposés être situés sur ou près d'une *forme géométrique* inconnue, le but est de construire un maillage approximant cette forme. Ainsi, vous pouvez voir le problème de la reconstruction comme celui de *relier les points dans le bon ordre pour découvrir la forme cachée*, un peu comme dans les jeux pour enfants, sauf qu'ici l'ordre entre les points n'est pas fourni... Formellement, par *forme géometrique* dans l'espace affine $\mathbb{R}^d$ on entendra simplement un compact de $\mathbb{R}^d$. La définition sera affinée dans la suite du chapitre, en fonction des besoins de notre analyse.

Les applications les plus évidentes de la reconstruction sont le rendu graphique et la simulation numérique. Les nuages de points fournis en entrée sont généralement obtenus à partir de scans d'objets physiques, comme illustré dans la figure 4.1. L'objectif est de bâtir des modèles numériques de ces objets qui soient simples à manipuler, tout en étant suffisamment informatifs sur les objets scannés. Les maillages sont d'excellentes représentations, gérées nativement et efficacement par les cartes graphiques modernes, et propices aux calculs de quantités différentielles nécessaires pour la simulation.

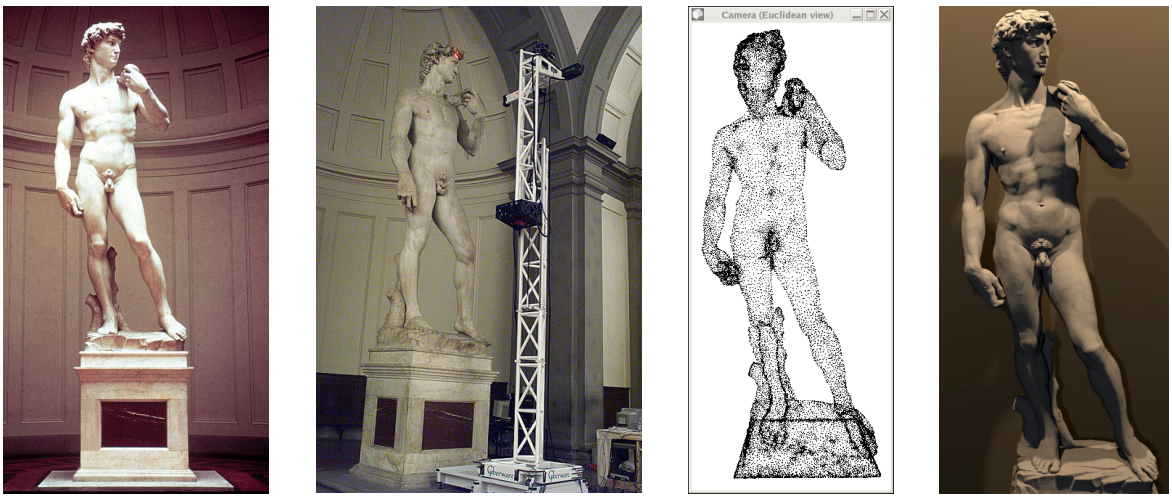


FIGURE 4.1 – *Processus de numérisation : de gauche à droite, la statue du David de Michel-Ange au musée de l'Académie à Florence, la disposition du scanner près de la statue, le nuage de points obtenu en sortie du scanner, et enfin le maillage reconstruit, rendu avec une texture uniforme dans une scène virtuelle. Ces images proviennent du* Stanford Michelangelo Project.

Une chose importante à bien comprendre est que la reconstruction est par nature un problème inverse, où l'on part d'un échantillonnage d'une forme et l'on veut retrouver cette forme, et qu'en cette qualité la reconstruction est un problème fondamentalement mal posé. Pour comprendre cette idée, regardons une version 1-dimensionnelle du problème, à savoir la reconstitution de signaux à partir d'échantillons. En toute généralité, une séquence d'échantillons donnée peut représenter une infinité de signaux différents, aux propriétés très diverses. Toutefois, si on se restreint à la classe des signaux de base sinusoïdale à largeur de bande bornée, alors le théorème de Shannon garantit que sous certaines conditions d'échantillonnage la séquence d'échantillons représente un unique signal qui peut être reconstitué.

La situation en reconstruction est exactement la même : par un nuage de points donné passent une infinité de formes géométriques aux propriétés géométriques et topologiques très différentes. Si l'on se restreint à une famille particulière de formes, qui dans la suite sera la famille des courbes et surfaces *à reach positif* (dont la définition sera donnée dans la section 4.1), alors on peut garantir que par le nuage ne passent que des formes équivalentes du point de vue géométrique et topologique. Tout comme dans le théorème de Shannon, cette propriété n'est vérifée que sous certaines conditions d'échantillonnage, qui stipulent que le nuage de points est suffisamment *dense* :

Définition 4.1. *Un* ε-échantillon *d'une forme géométrique* S *est un nuage de points* $P \subset S$ *tel que tout point* x *de* S *se situe à distance euclidienne au plus* ε *de* P*, i.e.* $\min_{p \in P} \|x - p\| \leq \varepsilon$.

L'intuition derrière l'ε-échantillonnage est claire : si certaines portions de S ne sont pas densément échantillonnées, alors la notion de *reconstruction correcte* est mal définie. Il est donc nécessaire que le nuage de points P soit dense partout sur S.

Dans ce chapitre nous allons parler de méthodes de reconstruction basées sur la triangulation de Delaunay, c'est-à-dire que le maillage approximant S sera extrait de $\mathrm{Del}(P)$, la triangulation de Delaunay du nuage de points P fourni en entrée. Plus précisément, nous allons définir un sous-complexe de $\mathrm{Del}(P)$ *canonique*, dont on montrera qu'il fournit une bonne approximation de S. Ce sous-complexe porte le nom de Delaunay restreint et est présenté dans la section 4.2. Il peut être calculé à partir des points de P seuls quand S est une courbe, comme nous le verrons dans la section 4.3. Quand S est une surface, on ne peut calculer le Delaunay restreint exactement : ceci n'est pas dû à un défaut de notre algorithme, mais plutôt à une limitation théorique liée à l'apparition de tétraèdres mal formés dans la triangulation de Delaunay, appelés *slivers*. Nous verrons cela dans la section 4.4, qui présente également quelques notes bibliographiques. Notez dès à présent que nous ne présentons qu'un seul algorithme de reconstruction parmi tant d'autres dans ce chapitre. Le choix de cet algorithme répond à trois critères importants pour ce cours : il est conceptuellement très simple, il vient avec des garanties théoriques, et il a un très bon comportement en pratique. Notez également qu'il est très récent (2007) et correspond grosso modo à l'état de l'art dans le domaine.

4.1 Reach d'une forme géométrique

Définition 4.2. *Étant donnée une forme géométrique* S *dans* $\mathbb{R}^d$*, l'axe médian* M_S *est l'adhérance du lieu des points de* $\mathbb{R}^d$ *qui ont au moins deux plus proches voisins sur* S.

En d'autres termes, l'axe médian peut être vu comme une sorte de squelette de la forme — voir les exemples de la figure 4.2. D'ailleurs, il coïncide avec une notion de squelette bien connue en géométrie et en traitement d'images :

Définition 4.3. *Le* squelette *d'une forme géométrique* S *est l'adhérence du lieu des centres de boules ouvertes maximales incluses dans* $\mathbb{R}^d \setminus S$.

Si l'inclusion de M_S dans le squelette de S est évidente (une boule ouverte vide ayant au moins deux points de contact avec S est forcément maximale dans le complémentaire), l'inclusion réciproque n'est pas évidente et découle en fait d'un résultat assez technique dû à Federer [Fed59, Thm 4.8(6)].

La distance à l'axe médian M_S est connue sous le nom de *local feature size* dans la littérature. Nous utiliserons indifféremment les deux termes dans la suite, avec comme notation lfs_S. Puisque S est compacte et lfs_S continue (en fait 1-Lipschitzienne), la restriction de la fonction lfs_S à S atteint son minimum, appelé *reach* de S et noté ϱ_S. Ce minimum ne peut être négatif puisque lfs_S est elle-même positive ou nulle.

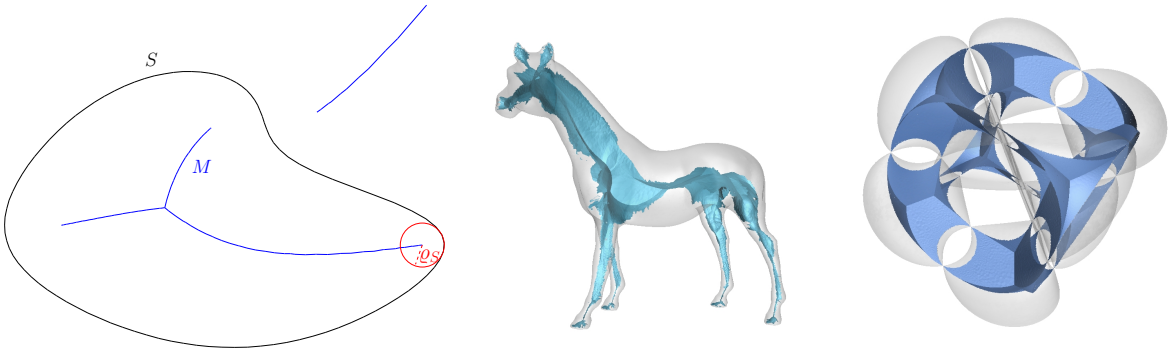


Figure 4.2 – *Quelques formes géométriques et (des approximations de) leurs axes médians. Pour des raisons évidentes de visualisation, sur les deux images de droite on ne montre que la partie de l'axe médian se trouvant à l'intérieur de la forme.*

Note historique. *La notion de reach a été introduite par Herbert Federer dans son article de 1959 sur les mesures de courbure [Fed59]. La situation à l'époque était la suivante : d'un côté la géométrie différentielle classique permettait d'analyser les propriétés locales et globales des formes lisses (comprendre : les compacts dont le bord est au moins deux fois différentiable) ; de l'autre, l'analyse convexe permettait d'étudier les propriétés des formes convexes sans requérir que les bords de ces formes soient différentiables partout. La contribution essentielle de Federer a été d'unifier les deux théories, en remarquant que les formes lisses et les formes convexes appartiennent en fait à un même ensemble aux propriétés suffisantes pour faire de l'analyse : celui des formes à reach positif. Il est à noter en effet que les formes à bord deux fois différentiable ont toujours un reach positif, tandis que les formes convexes ont un reach infini (et donc positif). Dans son article, Federer démontre que certains outils de la géométrie différentielle classique et de l'analyse convexe se généralisent naturellement aux cas des formes à reach positif. Par exemple, en tout point d'une forme à reach positif on peut définir un cône tangent et un cône normal, qui jouent des rôles équivalents à ceux joués par les espaces tangents et normaux dans le cas lisse. C'est ainsi qu'est née la théorie de l'analyse non lisse, développée et étendue plus tard par Cheeger [Che91] et Fu [Fu95], entre autres.*

Dans ce chapitre nous nous intéressons aux courbes et surfaces sans bord qui ont un reach positif. Il est connu depuis les travaux de Federer que ces dernières sont au moins une fois différentiables, et que leur champ de normales est Lipschitzien. Comme illustré dans la Figure 4.3 (centre), elles peuvent toutefois ne pas être de classe C^2, ce qui empêche l'utilisation de certains

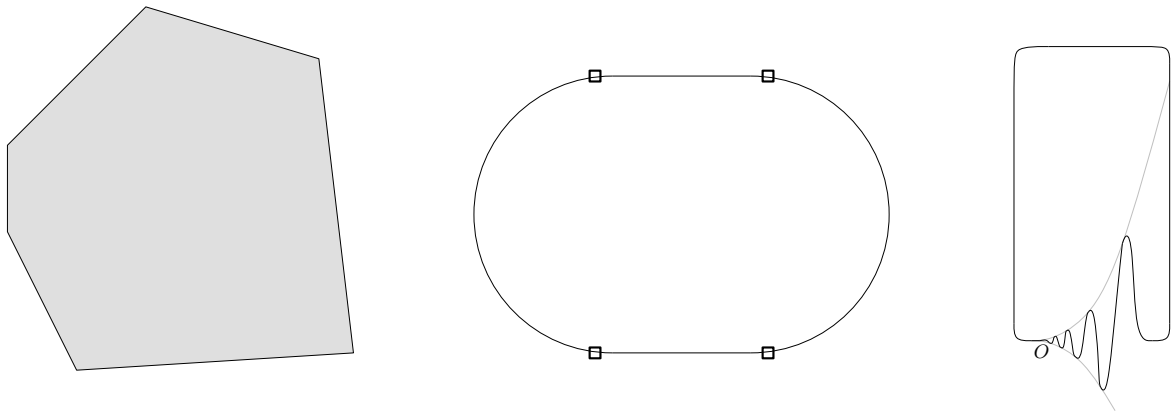


FIGURE 4.3 – *Quelques formes géométriques à reach positif ou nul. Le polygone convexe (plein) à gauche a un reach infini. La courbe au centre est formée de deux demi-cercles joints en leurs extrémités par des segments de droites horizontales. Aux points de jointure (marqués par des boîtes), la courbe n'est qu'une fois différentiable. Par contre, le champ des normales est Lipschitzien et donc le reach est positif (égal au rayon des cercles). La courbe de droite contient une portion qui coïncide avec le graphe de la fonction $x \mapsto x^3 \sin \frac{1}{x}$ (l'enveloppe $x \mapsto x^3$ est marquée en gris). Elle est une fois différentiable mais son champ de normales n'est pas Lipschitzien, et de fait l'axe médian vient toucher la courbe au point O, réduisant ainsi le reach à zéro.*

outils de géométrie différentielle classiques. Mais en fait nous n'en aurons pas besoin, et il s'avère qu'en nous appuyant sur les résultats de Federer nous allons produire des preuves plus simples et plus géométriques. Pour simplifier l'exposition, toutes les preuves seront données pour le cas des courbes. Le cas des surfaces est similaire, avec quelques adaptations toutefois, détaillées dans le cours de géométrie algorithmique de Master 2. Ces changements nécessitent d'utiliser des outils mathématiques plus élaborés comme la théorie de Morse [Mil63] et certains arguments de topologie différentielle [Hir76], qui requièrent une introduction minutieuse et sont donc laissés de côté ici.

Quelques lemmes techniques utiles. Dans la suite nous aurons besoin des résultats suivants, qui vont nous permettre de nous abstraire de la géométrie différentielle dans nos preuves :

Lemme 4.4 (Disques tangents)**.** *Soit S une courbe fermée dans le plan, avec un reach positif. Pour tout point x de S et tout $r \leq \mathrm{lfs}_S(x)$, les deux disques ouverts de rayon r tangents à S en x de part et d'autre de la courbe n'intersectent pas cette dernière.*

Démonstration. Soit $B(c, r)$ l'un de ces disques. Comme x se situe sur le cercle bordant le disque, il suffit de montrer que x est parmi les plus proches voisins de c sur S. Supposons par contradiction que ce ne soit pas le cas, et qu'au contraire il existe un point de S plus proche de c que ne l'est x. Notons d_S la distance à S, définie par $\mathrm{d}_S(p) = \min_{y \in S} \|p - y\|$, et considérons la fonction f qui à tout point p du segment de droite $[x, c]$ associe la quantité $\|p - x\| - \mathrm{d}_S(p)$. La valeur de cette fonction est évidemment nulle en x, et par hypothèse strictement positive en c. Comme la fonction est continue, l'ensemble $f^{-1}(\{0\})$ est un fermé de $[x, c]$, i.e. une union de sous-segments fermés de $[x, c]$ deux-à-deux disjoints. Soit $[x, p_0]$ l'élément de cette union qui contient x. Le disque ouvert D_0 centré en p_0 et de rayon $\|p_0 - x\|$ n'intersecte pas S, tandis que son adhérence touche S en x. Le seul moyen d'agrandir ce disque sans intersecter S est donc de déplacer son centre tout en restant tangent à D_0 en x. Or, par définition de p_0 il existe des points q sur le segment $[p_0, p]$ arbitrairement proches de p_0 tels que les disques ouverts centrés en ces points q et de rayon $\|q - x\|$ intersectent S. Il s'ensuit que D_0 est un disque maximal dans le

complémentaire de S. Dès lors, p_0 appartient au squelette de S et donc à M_S. Si $p_0 \neq c$, alors on conclut que $\mathrm{lfs}_S(x) \leq \|x - p_0\| < r \leq \mathrm{lfs}_S(x)$, ce qui contredit l'hypothèse du lemme. Si $p_0 = c$, alors on en déduit qu'aucun point de S n'est plus proche de c que ne l'est x, ce qui contredit l'hypothèse faite au début de la preuve. Dans tous les cas on conclut à une contradiction, ce qui prouve le lemme. □

La preuve de ce lemme reste exactement la même pour les surfaces dans $\mathbb{R}^3$, et même en fait pour les ensembles compacts en toutes dimensions.

Lemme 4.5 (Arc topologique)**.** *Soit S une courbe dans le plan à reach positif. Tout disque fermé dont l'intersection avec S n'est ni vide ni un arc (potentiellement dégénéré en un point) intersecte l'axe médian de S.*

Démonstration. Soit $D(c, r)$ un disque comme dans l'énoncé, c'est-à-dire que l'intersection $D(c, r) \cap S$ n'est ni vide ni un arc potentiellement dégénéré en un point. Ceci signifie que $D(c, r) \cap S$ a au moins deux composantes connexes distinctes $S_1, S_2 \subset S$. Si l'intersection a plus de composantes connexes, alors on prend pour S_1, S_2 les deux composantes qui passent au plus près de c. Soit x et y les minima globaux de la distance à c sur chacune de ces composantes, et supposons sans perte de généralité que $\|x-c\| \leq \|y-c\|$. S'il y a égalité entre les distances, alors c a deux plus proches voisins au moins sur S (soit x et y), ce qui fait que $c \in M_S$, et le lemme est prouvé. Si au contraire il n'y a pas égalité, alors la fonction $f(p) = \|p-y\| - \mathrm{d}_S(p)$, définie sur le segment de droite $[c, y]$, est nulle en y et strictement positive en $p = c$. Le même argument que dans la preuve du lemme 4.4 montre alors qu'il existe un point $p_0 \in [c, y]$ appartenant à M_S. Or, ce point est à distance au plus $\|c - y\| \leq r$ de c et donc appartient au disque $D(c, r)$. □

La preuve de ce lemme dans le cas des surfaces (et à fortiori dans celui des hypersurfaces) s'appuie non seulement sur des minima locaux de la fonction distance au point c, mais également sur des points critiques d'indices supérieurs au sens de la théorie de Morse. Toutefois, l'approche générale de la preuve reste la même, seuls les détails techniques se compliquent.

4.2 Triangulation de Delaunay restreinte

Le concept central utilisé dans la reconstruction basée sur la triangulation de Delaunay est la notion de restriction de la triangulation à une forme géométrique, définie ci-dessous et illustrée dans la figure 4.4 :

Définition 4.6. *Étant donné une forme S et un nuage de points $P \subset S$, le* diagramme de Voronoï restreint $\mathrm{Vor}_S(P)$ *est le complexe cellulaire défini par les faces de* $\mathrm{Vor}(P)$ *qui intersectent S. La triangulation de Delaunay restreinte* $\mathrm{Del}_S(P)$ *est le sous-complexe de* $\mathrm{Del}(P)$ *dual de* $\mathrm{Vor}_S(P)$*.*

Sous certaines hypothèses d'échantillonnage, le Delaunay restreint est une bonne reconstruction de la courbe ou de la surface sous-jacente au nuage de points :

Théorème 4.7 (Reconstruction par Delaunay restreint)**.** *Soit S une courbe fermée dans le plan (respectivement une surface sans bord dans l'espace), avec un reach positif, et soit P un ε-échantillon de S avec $\varepsilon < \varrho_S$ (respectivement $\varepsilon < 0.1\varrho_S$). Alors le complexe simplicial* $\mathrm{Del}_S(P)$ *est homéomorphe à S.*

Comme annoncé plus haut, nous allons démontrer ce résultat dans le cas des courbes, laissant le cas des surfaces et ses détails techniques un peu fastidieux de côté. L'idée de la preuve est très simple : d'une part, les arêtes du Delaunay restreint ne joignent que des points de P qui

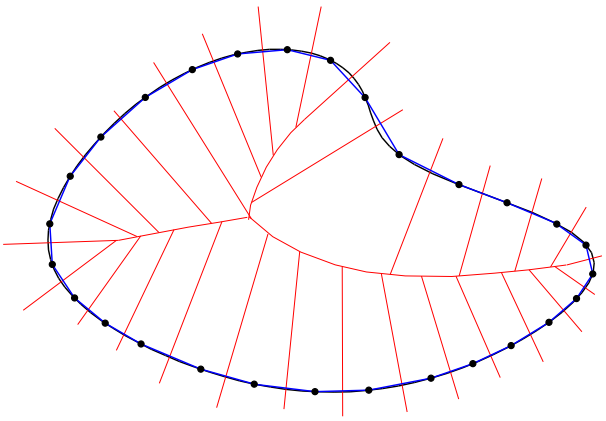


FIGURE 4.4 – *Delaunay restreint à une courbe plane. La courbe et le nuage de points sont en noir, les arêtes du diagramme de Voronoï en rouge, et les arêtes du Delaunay restreint en bleu. Il est à noter que le Delaunay restreint ne contient pas de triangles ici.*

sont consécutifs le long de la courbe (lemme 4.8) ; d'autre part, toutes les paires de points de P consécutifs le long de la courbe forment des arêtes dans le Delaunay restreint (lemme 4.9). Ainsi, le Delaunay restreint *suit* la courbe, comme le suggère l'exemple de la figure 4.4.

Lemme 4.8. *Soit P un ε-échantillon d'une courbe plane à reach positif. Si $\varepsilon < \varrho_S$, alors pour toute arête $e = [pq]$ de $\mathrm{Del}_S(P)$, les points p et q sont consécutifs le long de S, et de plus le(s) point(s) d'intersection entre S et l'arête de Voronoï duale de $[pq]$ est(sont) sur l'arc de S joignant p à q sans autre point de P entre.*

Démonstration. Soit P comme dans l'énoncé, et soient $p, q \in P$ tels que l'arête $[pq]$ est dans le Delaunay restreint. L'arête de Voronoï duale de $[pq]$ intersecte alors S, et soit c un point quelconque dans l'intersection : c est centre d'un disque D dont l'intérieur est vide de points de P, et tel que p, q sont sur le bord du disque. Il s'ensuit que la distance de c à P est précisément $\|c-p\| = \|c-q\|$, et comme P est un ε-échantillon de S, à laquelle c appartient, on déduit que $\|c-p\| = \|c-q\| \leq \varepsilon < \varrho_S \leq \mathrm{lfs}_S(c)$. D'où : le disque D n'intersecte pas l'axe médian de S, et par conséquent $D \cap S$ est un arc, d'après le lemme 4.5. Cet arc a ses sommets sur le cercle bordant D et contient p et q, eux aussi localisés sur le cercle. Si d'aventure l'arc contenait un troisième point s de P, alors s devrait également être localisé sur le cercle. Ainsi, l'arc aurait un point de tangence avec le cercle tel que de part et d'autre de ce point il resterait localisé à l'intérieur du disque. Il s'ensuivrait par le lemme 4.4 que le rayon de D est strictement plus grand que le reach de S et donc qu'ε, ce qui contredit l'hypothèse du lemme. Ainsi donc, l'arc de S inclus dans D ne contient pas d'autre point de S et a donc p et q comme extrémités, ce qui conclut la preuve. □

Lemme 4.9. *Soit P un ε-échantillon d'une courbe plane à reach positif. Si $\varepsilon < \varrho_S$, alors pour tout couple de points $p, q \in P$ consécutifs le long de S, l'arête $[pq]$ est une arête de $\mathrm{Del}_S(P)$.*

Démonstration. Soient S, p et q comme dans l'énoncé, et appelons $\overline{pq}$ l'arc de S joignant les points p et q sans autre point de P entre. Faisons bouger un point x le long de l'arc $\overline{pq}$, en partant de p : à un moment donné, x quitte la cellule de Voronoï de p pour entrer dans une autre cellule, disons celle d'un certain point $s \in P$. Si $s = q$, alors l'arête $[pq]$ est dans le Delaunay restreint. Sinon, c'est l'arête $[ps]$ qui est dans le Delaunay restreint, et par le lemme précédent (4.8) les points s et p doivent être consécutifs le long de S, avec x sur l'arc $\overline{sp}$, ce qui contredit notre hypothèse que $x \in \overline{pq}$. □

Maintenant que nous savons que le Delaunay restreint *suit* la courbe S, sans pour autant s'auto-intersecter puisqu'il est naturellement plongé dans le Delaunay 2-dimensionnel et que chaque point de P a exactement deux voisins distincts le long de la courbe S (car $\varepsilon < \varrho_S$), nous pouvons aisément construire un homémorphisme entre $\mathrm{Del}_S(P)$ et S. Pour cela, il nous suffit de considérer, pour chaque paire de points $p, q \in P$ consécutifs le long de S, un homéomorphisme entre l'arc $\overline{pq}$ et le segment de droite $[p, q]$ qui laisse les extrémités p et q invariantes. En recollant tous ces homéomorphismes locaux le long de leurs sommets communs on obtient un homéomorphisme global entre S et $\mathrm{Del}_S(P)$, ce qui complète la preuve du théorème 4.7.

4.3 Complexe de témoins et reconstruction multi-échelles

Revenons tout d'abord sur les difficultés inhérentes au problème de la reconstruction. Lorsqu'on admet du bruit dans les données, i.e. que les points du nuage d'entrée P ne sont pas exactement sur l'objet sous-jacent S, la notion même d'objet sous-jacent à P n'est plus clairement définie, et en fait elle dépend intimement de l'échelle à laquelle le nuage est observé. La figure 4.5 (gauche) en donne un exemple en 2D. Le même problème apparaît lorsque la dimension intrinsèque de l'objet recherché n'est pas connue a priori, comme illustré pour des courbes et surfaces en 3D dans la figure 4.5 (droite). Ainsi, même si dans un monde idéal où toutes les mesures sont exactes et les dimensions intrinsèques des objets sont connues on peut garantir que l'objet sous-jacent aux données est défini de manière unique (à équivalence près), dans le monde réel où les appareils de mesure commettent des erreurs (même infimes) et où le nombre de paramètres intrinsèques du système observé n'est pas forcément fourni (ceci est valable surtout en analyse de données et en apprentissage) on ne peut plus garantir formellement l'unicité de la reconstruction.

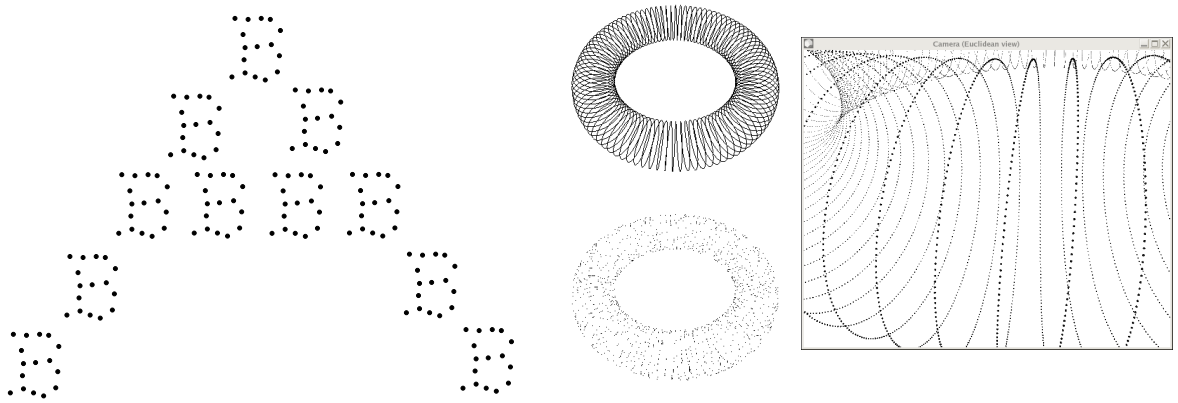


Figure 4.5 – *Quelques nuages de points représentant des structures multi-échelles. À gauche, le nuage représente une collection de B à petite échelle, mais un A bruité à plus grande échelle. À droite, le nuage est échantillonné uniformément le long d'une courbe hélicoïdale enroulée sur un tore, si bien que l'objet sous-jacent est la courbe à petite échelle et le tore à plus grande échelle.*

Alors que ce problème inhérent aux données est connu depuis longtemps des gens de l'apprentissage, ce n'est que très récemment qu'il a été pris en compte dans les travaux sur la reconstruction géométrique. La solution consiste à produire non pas un unique maillage à partir du nuage de points fourni en entrée, mais toute une famille de maillages paramétrée par l'échelle à laquelle chaque maillage approxime la forme sous-jacente à partir des données. C'est précisément ce que fait l'algorithme de Guibas et Oudot [GO08], dont le résultat sur le nuage

de la figure 4.5 (droite) est illustré dans la figure 4.6.

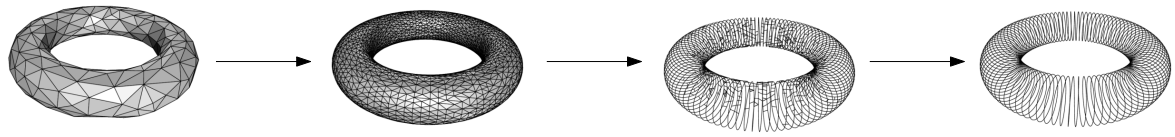


FIGURE 4.6 – *Extraits de la séquence de reconstructions produite par l'algorithme de Guibas et Oudot sur le nuage de la figure 4.5 (droite). De gauche à droite (i.e. de l'échelle la plus grande à l'échelle la plus petite), on découvre un tore grossier, puis un tore plus raffiné, puis un objet de transition entre le tore et la courbe, et enfin une approximation de la courbe.*

L'objectif de cette section sera de présenter une version basique de l'algorithme de Guibas et Oudot, et de montrer formellement qu'elle permet de reconstruire des courbes à différentes échelles. L'adaptation de l'algorithme aux surfaces (et plus généralement aux formes géométriques de toutes dimensions) est laissée de côté pour le Master 2, mais nous en dirons quelques mots dans la section 4.4. La notion d'échelle utilisée sera précisée au moment de la description de l'algorithme, dans la section 4.3.2. Mais tout d'abord nous allons présenter une nouvelle structure de données appelée *complexe de témoins*, qui peut être vue comme une version allégée de la triangulation de Delaunay, beaucoup plus facile à calculer et pourtant très proche.

4.3.1 Le complexe de témoins

Le complexe de témoins a été introduit par Vin de Silva [dS03, dS08] dans le but de remplacer le test coûteux de la boule vide, utilisé dans la construction de la trianglation de Delaunay, par un test beaucoup plus simple n'impliquant que des comparaisons de distances. C'est ce qu'il a appelé le test du témoin :

Définition 4.10. *Soient P et W deux ensembles finis de points dans $\mathbb{R}^d$.*
• *Un point $w \in W$ est* témoin *d'un simplex $\sigma = \{p_0, \cdots, p_k\} \subseteq P$ si les sommets de σ sont ses $k+1$ plus proches voisins parmi P, c'est-à-dire : $\forall q \in P$, $\forall i = 0, \cdots, k$, $\|w - p_i\| \leq \|w - q\|$.*
• *Le* complexe de témoins *$C_W(P)$ est le plus grand complexe simplicial formé de simplexes de P ayant des témoins dans W. Autrement dit, $C_W(P)$ est le complexe simplicial formé des simplexes dont toutes les faces (y-compris eux-mêmes) ont des témoins dans W.*

En gros, un témoin est un point du nuage W qui joue le rôle d'un centre de boule vide, ou du moins qui tente de le faire sachant qu'il n'est pas forcément équidistant des sommets du simplexe σ. On remarquera que $C_W(P)$ est a priori un complexe abstrait, dont l'image dans $\mathbb{R}^d$ pourrait éventuellement s'auto-intersecter. En réalité, le résultat suivant dû à de Silva [dS08] montre que $C_W(P)$ est un sous-complexe de la triangulation de Delaunay $\mathrm{Del}(P)$ et vit donc bien dans $\mathbb{R}^d$, comme illustré dans la figure 4.7 :

Théorème 4.11 (des témoins)**.** *Quels que soient les ensembles finis P et W, le complexe de témoins $C_W(P)$ est un sous-complexe de la triangulation de Delaunay* $\mathrm{Del}(P)$*. Plus précisément, pour tout simplexe σ de $C_W(P)$ il existe une boule de Delaunay circonscrite à σ et centrée dans l'enveloppe convexe des témoins de σ et de ses faces.*

La preuve originale de de Silva est un peu technique. Nous allons donc présenter une autre preuve, dûe à Attali et al. [AEM07], qui est très élégante et de nature très géométrique.

Démonstration. La preuve montre que chaque simplexe $\sigma = \{p_0, \cdots, p_k\}$ de $C_W(P)$ est un simplexe de Delaunay, et elle procède par une double induction : d'une part sur la dimension

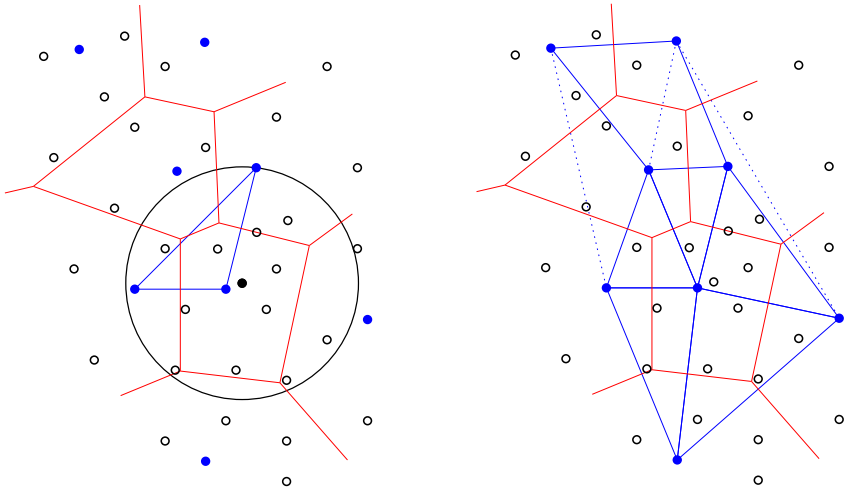


Figure 4.7 – *Gauche : le point noir plein est témoin du triangle bleu. Droite : le complexe de témoins des points bleus avec comme ensemble de témoins l'ensemble noir. Le complexe est inclus dans la triangulation de Delaunay des points bleus, dont les arêtes manquantes sont figurées en pointillés. Notez que le triangle de la figure de gauche n'est pas dans le complexe parce qu'une de ses arêtes n'a pas de témoin, bien que lui-même ait un témoin.*

k de σ, et d'autre part sur la taille d'un sous-ensemble particulier des sommets de σ que nous identifierons en temps voulu. Commençons par l'induction principale :

- Cas $k = 0$: les sommets de P étant des points de P, ils sont tous automatiquement sommets de $\mathrm{Del}(P)$.

- Cas $k > 0$: Supposons que tout simplexe de $C_W(P)$ de dimension $l < k$ est inclus dans $\mathrm{Del}(P)$, et montrons qu'alors tout simplexe $\sigma = \{p_0, \cdots, p_k\}$ de $C_W(P)$ de dimension k est également dans $\mathrm{Del}(P)$. Pour un tel simplexe il existe un témoin $w_\sigma \in W$, centre d'une boule B_σ séparant $p_0, \cdots, p_k$ des autres points de P. Nous pouvons maintenant débuter l'induction auxiliaire :

• Si aucun p_i ne se trouve sur le bord de la boule, alors nous pouvons diminuer le rayon de B_σ de manière à ce que l'un des p_i au moins (disons p_0 sans perte de généralité) soit sur le bord de B_σ tandis que cette dernière sépare toujours $p_0, \cdots, p_k$ du reste de P.

• Supposons maintenant sans perte de généralité que $p_0, \cdots, p_l$ se trouvent sur le bord de la boule (modifée) B_σ tandis que $p_{l+1}, \cdots, p_k$ se situent à l'intérieur. Si $l = k$, alors σ est un simplexe de Delaunay. Sinon, le simplexe $\tau = \{p_0, \cdots, p_l\}$ est une face propre de σ, de dimension $l < k$, donc notre hypothèse de récurrence implique que τ est un simplexe de Delaunay, i.e. qu'il existe une boule B_τ circonscrite à τ dont l'intérieur est vide de points de P. Soit c_τ le centre de B_τ. Comme les boules B_τ et B_σ sont toutes deux circonscrites à $p_0, \cdots, p_l$, le segment de droite $[c_\tau, w_\sigma]$ est lieu des centres d'un faisceau de boules circonscrites à $p_0, \cdots, p_l$. En déplaçant w_σ le long de $[w_\sigma, c_\tau]$ dans la direction de c_τ, on se déplace dans le faisceau de boules circonscrites jusqu'à obtenir une boule B' contenant toujours $p_0, \cdots, p_k$ mais dont le bord contient un point p_i supplémentaire, que nous supposerons être p_{l+1} sans perte de généralité. Ainsi nous avons identifié une boule B' circonscrite à $p_0, \cdots, p_l, p_{l+1}$, qui contient $p_{l+2}, \cdots, p_k$, et qui, étant localisée dans le faisceau, est incluse dans $B_\sigma \cup B_\tau$, ce qui fait que son intérieur ne contient aucun autre point de P.

En déroulant l'induction auxiliaire nous concluons qu'il existe une boule de Delaunay circonscrite à σ, qui se trouve donc dans $\mathrm{Del}(P)$. De plus, la manière dont nous avons construit cette boule fait que son centre est inclus dans l'enveloppe convexe des témoins de σ et de ses faces. Ceci termine l'induction principale et par là même la preuve du théorème. □

4.3.2 L'algorithme de reconstruction multi-échelles

Soit W un nuage de points dans $\mathbb{R}^d$. En fait, la construction du complexe de témoins ne nécessitant que de comparer des distances, l'algorithme peut être utilisé dans n'importe quel espace métrique. L'idée de base de l'algorithme est la suivante : on construit itérativement un sous-ensemble P de points de W, en commençant par $P = \emptyset$ et en ajoutant un nouveau point dans P à chaque itération, jusqu'à ce que $P = W$ (condition de terminaison de l'algorithme). Dans le même temps, on maintient le complexe de témoins du sous-ensemble P en prenant tous les points de W comme témoins. La sortie de l'algorithme est la séquence des complexes de témoins ainsi construits.

Algorithm 5 *Version basique de l'algorithme de Guibas et Oudot.*

`Entrée :` W un nuage de points dans $\mathbb{R}^d$.
Soit $P := \emptyset$;
Soit $C_W(P) := \emptyset$;
Soit $\Sigma = [\,]$; *// la séquence vide*
`Tant que` $P \neq W$
  Soit $w^* := \text{argmax}_{w \in W} \min_{p \in P} \|w - p\|$; *// w^* est choisi arbitrairement quand $P = \emptyset$*
  $P := P \cup \{w^*\}$;
  Mettre à jour $C_W(P)$;
  Ajouter $C_W(P)$ à la séquence Σ ;
`Sortie :` la séquence Σ.

Le pseudo-code est fourni dans l'algorithme 5. Notez que la mise à jour du complexe de témoins $C_W(P)$ après insertion dans P n'est pas détaillée ici, pour simplifier l'exposition. Notez également que le choix du nouveau point à insérer dans P à chaque itération est fait selon un critère de distance : c'est le point de W le plus éloigné de P qui est inséré. Ceci garantit que le sous-ensemble P échantillonne le nuage original W uniformément. En effet, en appelant $\lambda(i)$ la distance $\max_{w \in W} \min_{p \in P} \|w - p\|$ calculée au début de la i-ème itération de l'algorithme, on a la propriété suivante :

Lemme 4.12. *À la fin de la i-ème itération de l'algorithme, P est un $\lambda(i)$-échantillon de W, et de plus il est $\lambda(i)$-*creux*, i.e. les distances entre points distincts de P sont toutes supérieures ou égales à $\lambda(i)$.*

Démonstration. La définition même de $\lambda(i)$ implique que P est un $\lambda(i)$-échantillon de W au début de l'itération i. Comme P grossit au cours de cette itération, il reste un $\lambda(i)$-échantillon de W. Maintenant, comme P ne fait que grossir au cours de l'exécution de l'algorithme, la quantité λ ne fait que diminuer. Ainsi, pour chaque iteration $j \leq i$, le point inséré dans P se trouve à distance au moins $\lambda(j) \geq \lambda(i)$ de P lors de son insertion. Ceci implique que les points de P sont à distance au moins $\lambda(i)$ les uns des autres à la fin de l'itération i. □

Les premières itérations de l'algorithme sont illustrées dans la figure 4.8 sur un exemple de nuage échantillonnant une courbe simple fermée dans le plan avec du bruit. On note que la topologie de la courbe est rapidement capturée par le complexe de témoins. Ce dernier garde cette topologie pendant un long moment, jusqu'à ce que finalement le sous-ensemble P devienne trop dense par rapport à l'ensemble des témoins W pour que $C_W(P)$ continue d'être une courbe fermée. À la fin du processus, lorsque $P = W$, la topologie de $C_W(P)$ devient celle du graphe reliant tout point de W à son plus proche voisin, et elle n'a donc plus rien à voir avec celle de la courbe, comme illustré dans la partie droite de la figure 4.8.

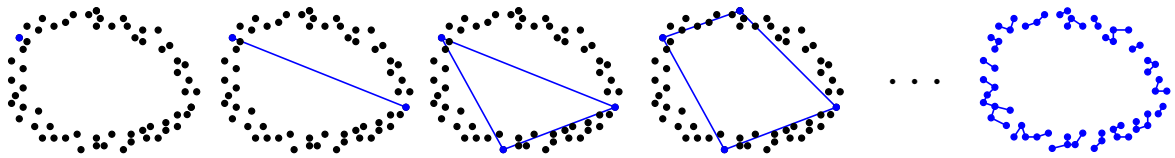


Figure 4.8 – *Illustration des premières étapes de l'algorithme. Le sous-ensemble P et son complexe de témoins sont marqués en bleu. Le premier point bleu est choisi arbitrairement parmi W. À la fin du processus (à droite), on a $P = W$ et $C_W(P)$ devient le graph de plus proche voisin de W.*

Ainsi, au sein de la séquence de reconstructions construite par l'algorithme, une longue sous-séquence approxime fidèlement la courbe sous-jacente aux données. Cette observation est confirmée par le résultat théorique suivant, dont la preuve sera l'objet de la prochaine section :

Théorème 4.13 (Reconstruction par complexe de témoins)**.** *Soient S une courbe plane fermée à reach positif, W un ε-échantillon de S, et P un λ-échantillon de W qui est λ-creux. Si $7\varepsilon \leq \lambda < \frac{1}{4}\varrho_S - \varepsilon$, alors $C_W(P)$ coïncide avec $\mathrm{Del}_S(P)$, qui d'après le théorème 4.7 est homéomorphe à S.*

Ce théorème garantit qu'à toute itération i de l'algorithme telle que $\lambda(i)$ se situe entre 7ε et $\frac{1}{4}\varrho_S - \varepsilon$, on a la garantie que le complexe de témoins $C_W(P)$ capture la topologie de la courbe S. La plage de valeurs admissibles pour $\lambda(i)$ a une largeur de $\frac{1}{4}\varrho_S - 8\varepsilon$ et dépend donc directement de la densité d'échantillonnage du nuage de points W fourni en entrée : plus cette densité est grande (i.e. plus ε est petit), plus la plage est grande et donc plus on a de chances de pouvoir détecter une (longue) sous-séquence de reconstructions correspondant à la courbe S dans la sortie de l'algorithme.

Comment détecter de telles sous-śequences en pratique ? Il suffit pour cela de calculer des invariants topologiques comme les nombres de Betti, puis d'afficher le graphe d'évolution de ces invariants au fur et à mesure de l'algorithme. C'est ce que nous avons fait dans la figure 4.9 (gauche) avec les deux premiers nombres de Betti (le premier représente le nombre de composantes connexes, le deuxième le nombre de cycles), dont nous avons tracé l'évolution en fonction de la quantité (croissante) $1/\lambda(i)$. Cette représentation des invariants sous forme de diagramme dans le plan permet à l'utilisateur de facilement détecter les plages où la topologie de la reconstruction est stable.

4.3.3 Correction de l'algorithme

Le but de cette sous-section est de démontrer formellement le théorème 4.13. Elle peut donc être ignorée lors d'une première lecture. La preuve du théorème requiert les résultats techniques suivants :

Lemme 4.14 (de la corde)**.** *Soit S une courbe plane à reach positif, et soient p, q deux points sur cette courbe. Si $\|p - q\| < 2\varrho_S$, alors l'angle entre la droite (p, q) et la droite tangente à S en p est au plus* $\arcsin \frac{\|p-q\|}{2\varrho_S}$.

Démonstration. Soient S, p et q comme ci-dessus, et soit C le cercle de centre p et rayon $\|p-q\|$. Comme $\|p - q\| < 2\varrho_S$, ce cercle intersecte les deux cercles de rayon ϱ_S tangents à S en p en quatre points exactement, un dans chaque quadrant délimité par les droites tangente et normale à S en p — voir la figure 4.10. Soit s' le point d'intersection se trouvant dans le même quadrant que q. Comme les intérieurs des deux cercles tangents à S en p sont vides de points de S, q se situe à l'extérieur et donc l'angle entre la droite (p, q) et la droite tangente à S en p est

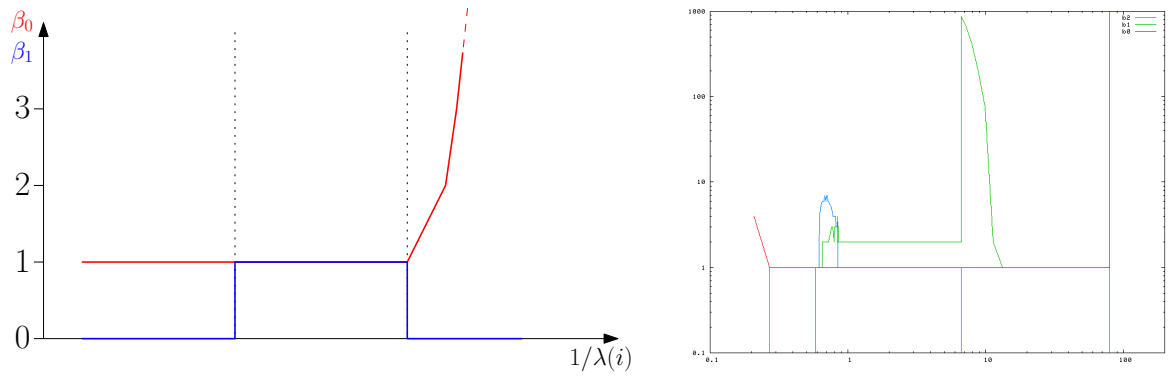


FIGURE 4.9 – *Diagramme d'évolution des nombres de Betti dans la séquence de reconstructions bâtie par l'algorithme. À Gauche, diagramme de la séquence obtenue par l'algorithme basique sur le nuage de la figure 4.8. À droite, diagramme de la séquence obtenue par une version plus élaborée de l'algorithme sur le nuage de la figure 4.5 (droite).*

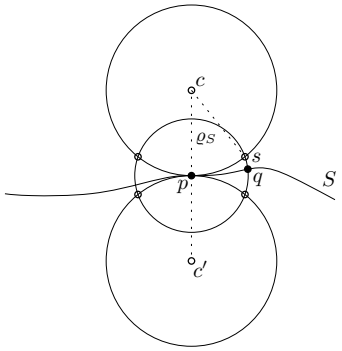


FIGURE 4.10 – *Pour la preuve du lemme 4.14.*

au plus l'angle α entre cette dernière et (p, s). En appelant c le centre du cercle tangent du même côté de la tangente à S en p que s, on a que α est égal au demi-angle issu de c dans le triangle isocèle (p, c, s), qui vaut $\arcsin \frac{\|p-s\|}{2\varrho_S} = \arcsin \frac{\|p-q\|}{2\varrho_S}$ par un calcul simple (voir encore la figure 4.10). □

Lemme 4.15 (des sommets)**.** *Soit S une courbe plane à reach positif, et soit P un nuage de points échantillonnés sur cette courbe. Alors tous les sommets de* $\mathrm{Vor}(P)$ *sont à distance au moins $\frac{\varrho_S}{2}$ de P.*

Démonstration. Soit c un sommet de $\mathrm{Vor}(P)$ et soient p, q, s les trois sommets du simplexe de Delaunay dual de c. Considérons la boule B de centre c et de rayon $\|c-p\| = \|c-q\| = \|c-s\|$. Le bord de cette boule contient les points p, q, s, qui eux-mêmes appartiennent à S, donc deux possibilités s'offrent à nous :

- soit $B \cap S$ n'est pas connexe, auquel cas le lemme 4.5 implique que $B \cap M_S \neq \emptyset$, ce qui implique que $\|c-p\| \geq \frac{1}{2}\mathrm{lfs}_S(p) \geq \frac{\varrho_S}{2}$,
- soit l'arc $B \cap S$ intersecte tangentiellement le cercle bordant B en un point $x \in \{p, q, s\}$ tel que S est inclus dans B de part et d'autre de x, de sorte qu'on a $\|c-x\| \geq \mathrm{lfs}_S(x) \geq \varrho_S$ par le lemme 4.4.

Dans tous les cas la distance de c à P (qui est aussi le rayon de B) est au moins égale à $\frac{\varrho_S}{2}$. □

Lemme 4.16 (de la distance)**.** *Soit P un $(\lambda+\varepsilon)$-échantillon d'une courbe plane à reach positif. Si $\lambda+\varepsilon < \varrho_S$, alors la distance de tout point de S à son deuxième plus proche voisin dans P est au plus $2(\varepsilon+\lambda)$.*

Démonstration. Soit $x \in S$ et soient p, q les deux points de P consécutifs le long de S situés de part et d'autre de x. D'après le lemme 4.9, l'arête $[pq]$ est dans $\mathrm{Del}_S(P)$. Soit $c \in S$ le centre d'une boule de Delaunay circonscrite à $[pq]$. Comme P est un $(\lambda+\varepsilon)$-échantillon de S, on a $\|c-p\| = \|c-q\| \leq \lambda+\varepsilon$. Maintenant, par hypothèse le point x se situe sur l'arc de courbe $\overline{pq}$, qui lui-même se situe dans la boule de centre c et de rayon $\|c-p\|$, comme on l'a vu dans la preuve du lemme 4.8. Il s'ensuit que $\|x-p\| \leq 2\|c-p\| \leq 2(\lambda+\varepsilon)$ et $\|x-q\| \leq 2\|c-q\| \leq 2(\lambda+\varepsilon)$, ce qui implique que la distance de x à son deuxième plus proche voisin dans P est au plus $2(\lambda+\varepsilon)$. □

Nous pouvons maintenant donner la preuve du théorème 4.13, qui procède par double inclusion : d'une part $C_W(P) \subseteq \mathrm{Del}_S(P)$ (lemme 4.17), d'autre part $\mathrm{Del}_S(P) \subseteq C_W(P)$ (lemme 4.18).

Lemme 4.17. *Sous les hypothèses du théorème 4.13, $C_W(P)$ est inclus dans* $\mathrm{Del}_S(P)$.

Démonstration. Tout d'abord, les sommets de $C_W(P)$ sont des points de P, qui par définition sont tous sommets de $\mathrm{Del}_S(P)$.

Regardons maintenant le cas d'une arête $[pq]$ de $C_W(P)$. Grâce au lemma 4.8, il nous suffit de montrer que p, q sont des points de P consécutifs le long de S. Soit $w \in W$ un témoin de $[pq]$. En supposant sans perte de généralité que $\|w-p\| \geq \|w-q\|$, la boule B_w de centre w et de rayon $\|w-p\|$ sépare p, q du reste de P. En bougeant w le long du segment de droite $[wp]$ dans la direction de p, on atteint un point c qui est équidistant de p et q. La boule B_c de centre c et de rayon $\|c-p\|$ étant incluse dans B_w, son intérieur est vide de points de P et c'est donc une boule de Delaunay. Maintenant, comme p est parmi les deux plus proches voisins de w dans P, le lemme 4.16 implique que $\|p-w\| \leq 2(\lambda+\varepsilon)$. Comme $c \in [pw]$, on déduit que $\|p-c\| \leq \|p-w\| \leq 2(\lambda+\varepsilon)$. Les points de la boule B_c sont donc tous à distance au plus $4(\lambda+\varepsilon)$ de p. Comme par hypothèse on a $4(\lambda+\varepsilon) < \varrho_S$, B_c n'intersecte pas l'axe médian M_S et donc intersecte S selon un arc topologique d'extrémités p, q. Le même argument que dans la preuve du lemme 4.8 montre alors que cet arc ne peut contenir d'autre point de P, ce qui implique que p et q sont consécutifs le long de S.

Reste le cas des simplexes de $C_W(P)$ de dimension plus grande. En fait, $C_W(P)$ ne peut contenir de triangles. En effet, nous venons de montrer que toute arête de $C_W(P)$ joint des points de P consécutifs le long de S. Le seul moyen pour que les trois arêtes d'un triangle $[pqs]$ soient dans $C_W(P)$ est donc que p, q, s soient les seuls points de P sur la composante connexe de S les contenant. Soit alors p' le point de S le plus éloigné de p : la boule de centre p et de rayon $\|p-p'\|$ contient toute la courbe S, qui est sans bord et donc n'a pas la topologie d'un arc. D'où, cette boule doit intersecter l'axe médian M_S, d'après le lemme 4.5. Il s'ensuite que $\|p-p'\| \geq \mathrm{lfs}_S(p) \geq \varrho_S$. Maintenant supposons sans perte de généralité que q est le point de P le plus proche de p'. Comme P est un $(\lambda+\varepsilon)$-échantillon de S, on a $\|q-p'\| \leq \lambda+\varepsilon$. D'où, $\|p-q\| \geq \varrho_S - (\lambda+\varepsilon)$. Enfin, p et q étant consécutifs le long de S, le lemme 4.9 implique que $[pq]$ est une arête de $\mathrm{Del}_S(P)$, circonscrite par au moins une boule de Delaunay centrée sur S. En appelant c le centre de cette boule, on a $\|p-q\| \leq \|p-c\| + \|c-q\| \leq 2(\lambda+\varepsilon)$. En conclusion, pour que le triangle $[pqs]$ soit dans $C_W(P)$ il faut que $2(\lambda+\varepsilon) \geq \varrho_S - (\lambda+\varepsilon)$, i.e. $\lambda+\varepsilon \geq \frac{1}{3}\varrho_S$, ce qui contredit l'hypothèse du lemme. Le complexe de témoin ne contient donc aucun triangle, et de ce fait aucun simplexe de dimension plus grande. □

Lemme 4.18. *Sous les hypothèses du théorème 4.13,* $\mathrm{Del}_S(P)$ *est inclus dans $C_W(P)$.*

Démonstration. Tout d'abord, les sommets de $\mathrm{Del}_S(P)$ sont des points de P, et comme P est inclus dans W, ils sont leurs propres témoins. Ainsi, les sommets de $\mathrm{Del}_S(P)$ sont dans $C_W(P)$.

Regardons maintenant le cas d'une arête $[pq] \in \mathrm{Del}_S(P)$. Soit $c \in S$ le centre d'une boule de Delaunay circonscrite à $[pq]$. On a alors $\|c-p\| = \|c-q\| \leq \lambda+\varepsilon$ puisque P est un $(\lambda+\varepsilon)$-échantillon de S. Et comme W est un ε-échantillon de S, il existe un point $w \in W$ à distance au plus ε de c. Nous allons montrer que w est un témoin de $[pq]$, i.e. que p et q sont les deux plus proches voisins de w parmi les points de P. Supposons le contraire, c'est-à-dire qu'il existe un point $s \in P \setminus \{p,q\}$ qui est plus proche de w que ne l'est p ou q. Les points p, q, s sont alors tous trois inclus dans la boule B_w de centre w et de rayon $\lambda+2\varepsilon$. En appelant respectivement p', q' et s' leurs projections orthogonales sur la droite T_w tangente à S en w, on a d'après le lemme de la corde (4.14) que les distances $\|p-p'\|$, $\|q-q'\|$ et $\|s-s'\|$ sont au plus $\frac{(\lambda+2\varepsilon)^2}{2\varrho_S}$. Supposons sans perte de généralité que, sur la droite T_w, les points q' et s' sont du même côté de w (le cas où p' et s' sont du même côté est identique, et le cas où p' et q' sont du même côté est similaire avec une analyse plus simple). Comme w, q', s' sont alignés, on a :

$$\begin{aligned}
\|q-s\|^2 &\leq (\|q-q'\| + \|s'-s\|)^2 + \|q'-s'\|^2 \\
&= (\|q-q'\| + \|s'-s\|)^2 + (\|w-s'\| - \|w-q'\|)^2 \\
&\leq \tfrac{(\lambda+2\varepsilon)^4}{\varrho_S^2} + \big|\|w-s'\|^2 - \|w-q'\|^2\big| \\
&\leq \tfrac{5(\lambda+2\varepsilon)^4}{4\varrho_S^2} + \big|\|w-s\|^2 - \|w-q\|^2\big| = \tfrac{5(\lambda+2\varepsilon)^4}{4\varrho_S^2} + (\|w-s\| + \|w-q\|)\,|\|w-s\| - \|w-q\|| \\
&\leq \tfrac{5(\lambda+2\varepsilon)^4}{4\varrho_S^2} + 2(\lambda+2\varepsilon)\,|\|w-s\| - \|w-q\||\,.
\end{aligned}$$

Or, comme P est λ-creux par hypothèse, on a $\|q-s\|^2 \geq \lambda^2$, ce qui implique que

$$|\|w-s\| - \|w-q\|| \geq \frac{1}{2(\lambda+2\varepsilon)} \left(\lambda^2 - \frac{5(\lambda+2\varepsilon)^4}{4\varrho_S^2}\right),$$

qui est strictement supérieur [1] à 2ε sous les hypothèses du théorème 4.13, à savoir $\varepsilon \leq \frac{\lambda}{7}$ et $\lambda+\varepsilon < \frac{1}{4}\varrho_S$. Dès lors, si $\|w-s\| \geq \|w-q\|$ alors on a également $\|w-s\| - \|w-p\| \geq \|w-s\| - \|w-q\| - 2\varepsilon \geq 0$, ce qui contredit l'hypothèse que s est plus proche de w que ne l'est p ou q. Sinon ($\|w-s\| \leq \|w-q\|$), on a $\|c-s\| \leq \|c-w\| + \|w-s\| < \varepsilon + \|w-q\| - 2\varepsilon \leq \|c-q\|$, ce qui contredit le fait que c est centre d'une boule de Delaunay circonscrite à l'arête $[pq]$. Dans tous les cas on tombe sur une contradiction, ce qui implique que w est effectivement témoin de l'arête $[pq]$. Comme d'après ce qui précède les points p, q sont sommets de $C_W(P)$, l'arête elle-même est dans $C_W(P)$.

Reste le cas des simplexes de dimension plus grande. En fait, $\mathrm{Del}_S(P)$ ne contient aucun triangle et donc aucun simplexe de dimension supérieure. En effet, d'après le lemme 4.15, les sommets de Voronoï duaux des triangles de Delaunay se situent à distance au moins $\frac{\varrho_S}{2}$ de P et n'intersectent donc pas S puisque P est un $(\lambda+\varepsilon)$-échantillon de S avec $\lambda+\varepsilon < \frac{\varrho_S}{2}$. □

Ceci conclut la preuve du théorème 4.13 et démontre ainsi la correction de notre algorithme.

4.4 Extensions et notes bibliographiques

Nous avons volontairement restreint notre analyse au cas des courbes planaires, afin de simplifier l'exposition et de rendre nos arguments plus intuitifs. Il est toutefois important de noter que l'approche de la reconstruction basée sur la triangulation de Delaunay fonctionne en toutes dimensions, modulo quelques adaptations :

1. Les calculs ont été vérifiés avec Maple, qui donne $\varepsilon \leq 0.157\lambda$ et $\lambda+\varepsilon < \frac{1}{4}\varrho_S$ comme condition suffisante.

• Comme indiqué dans le théorème 4.7, le résultat d'approximation des formes géométriques échantillonnées par le Delaunay restreint s'étend directement au cas des surfaces dans $\mathbb{R}^3$, à une constante près. Ce résultat, démontré par N. Amenta et M. Bern en 1999 [AB99], a été véritablement le point de départ de la théorie de la reconstruction basée sur la triangulation de Delaunay. Depuis, toute une famille d'algorithmes (le Crust, le Power Crust, le Cocone, les voisins naturels, etc.) ont été proposés, qui viennent tous avec des garanties théoriques sur la qualité de la reconstruction — voir par exemple [CG06] pour un survey. Le défaut commun à tous ces algorithmes est de ne considérer les données qu'à une seule échelle. En cela, la méthode présentée ici est particulièrement novatrice.

• Le résultat d'égalité entre Delaunay restreint et complexe de témoins, énoncé dans le théorème 4.13, n'est plus valide sur les surfaces. Plus précisément, alors que le complexe de témoins est toujours inclus dans le Delaunay [AEM07], il peut maintenant contenir des trous dûs à la présence de tétraèdres mal formés dans partie de la triangulation de Delaunay 3D se trouvant proche de la surface — voir l'exemple de la figure 4.11. Ces trous empêchent le complexe de témoins de contenir tout le Delaunay restreint. Pour pallier ce problème, on enrichit le complexe en relaxant le test du témoin donné dans définition 4.10, de manière à inclure à nouveau le Delaunay restreint [GO08].

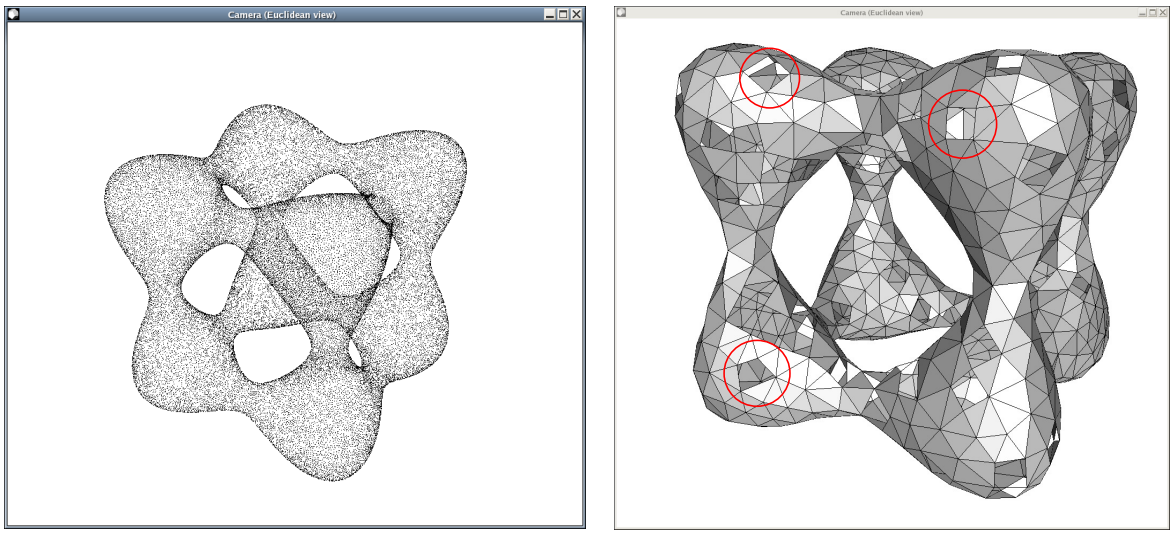


Figure 4.11 – *À gauche, un nuage de points échantillonnés sur une surface dans $\mathbb{R}^3$. À droite, le complexe de témoins bâti sur un sous-ensemble fixe du nuage, qui comporte des trous (entourés en rouge sur la figure) quelle que soit la densité du nuage.*

• Les résultats ci-dessus ne sont plus du tout valides en dimension plus grande que 3. Déjà sur les hypersurfaces dans $\mathbb{R}^4$, le complexe de témoins ne contient ni n'est contenu dans le Delaunay restreint, qui lui-même n'est pas forcément homéomorphe à l'hypersurface [Oud06]. Pour pallier ce problème il a été proposé par Cheng et al. [CDR05] d'utiliser des triangulations régulières au lieu de triangulations de Delaunay : par un choix judicieux des poids affectés aux sommets, on peut garantir que le Delaunay restreint est homéomorphe à la forme géométrique sous-jacente. Cette idée a été reprise par Boissonnat et al. [BGO09] pour montrer que dans ces conditions le complexe de témoins coïncide à nouveau avec le Delaunay (à poids) restreint. Ainsi il est possible de faire de la reconstruction multi-échelle à partir de nuages de points en toutes dimensions. Notez toutefois que la complexité de l'approche devient vite prohibitive : pour n points dans $\mathbb{R}^d$ fournis en entrée, l'algorithme utilise un espace mémoire de l'ordre de $d^{O(d^2)}n$ et a une durée d'exécution de l'ordre de $d^{O(d^2)}n^2$, des quantités qui deviennent véritablement astronomiques

dès que la dimension d dépasse 4 ou 5. Ainsi, en l'état actuel des choses, l'approche multi-échelles proposée par Guibas et Oudot [GO08] et adaptée en toutes dimensions par Boissonnat et al. [BGO09] demeure purement théorique en dimensions moyennes et grandes (typiquement pour d supérieur à 4 ou 5). D'autres approches plus légères ont vu le jour récemment : basées sur la théorie de la *persistance topologique* [ELZ02, ZC05], ces méthodes ne font pas exactement de la reconstruction mais sont capables d'inférer des invariants topologiques non triviaux d'une forme échantillonnée en un temps qui reste raisonnable. C'est une direction de recherche très prometteuse, en plein essor actuellement.

Bibliographie

[AB99] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete Comput. Geom.*, 22(4) :481–504, 1999.

[AEM07] D. Attali, H. Edelsbrunner, and Y. Mileyko. Weak witnesses for Delaunay triangulations of submanifolds. In *Proc. ACM Sympos. on Solid and Physical Modeling*, pages 143–150, 2007.

[BGO09] J.-D. Boissonnat, L. J. Guibas, and S. Y. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. *Discrete and Computational Geometry*, 42(1) :37–70, 2009.

[CDR05] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *Proc. 16th Sympos. Discrete Algorithms*, pages 1018–1027, 2005.

[CG06] F. Cazals and J. Giesen. Delaunay triangulation based surface reconstruction. In J.D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 231–273. Springer, 2006.

[Che91] J. Cheeger. Critical points of distance functionsand applications to geometry. Number 1504 in Springer Lecture Notes, pages 1–38, 1991.

[dS03] V. de Silva. A weak definition of Delaunay triangulation. Technical report, Stanford University, October 2003.

[dS08] V. de Silva. A weak characterisation of the Delaunay triangulation. *Geometriae Dedicata*, 135(1) :39–64, August 2008.

[ELZ02] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28 :511–533, 2002.

[Fed59] H. Federer. Curvature measures. *Trans. Amer. Math. Soc.*, 93 :418–491, 1959.

[Fu95] J. H. G. Fu. Tubular neighborhoods in Euclidean spaces. *Duke Math. Journal*, 52(4), 1995.

[GO08] L. J. Guibas and S. Y. Oudot. Reconstruction using witness complexes. *Discrete and Computational Geometry*, 40 :325–356, 2008.

[Hir76] M. W. Hirsch. *Differential Topology.* Springer-Verlag, New York, NY, 1976.

[Mil63] J. W. Milnor. *Morse Theory.* Princeton University Press, Princeton, NJ, 1963.

[Oud06] S. Y. Oudot. On the topology of the restricted Delaunay triangulation and witness complex in higher dimensions. Technical report, Stanford University, November 2006. LANL arXiv :0803.1296v1 [cs.CG], `http://arxiv.org/abs/0803.1296`.

[ZC05] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2) :249–274, 2005.

Chapitre 5

Problèmes de proximité

5.1 Localisation planaire en petite dimension

Nous avons déjà défini et abordé le problème de localisation dans une subdivision planaire au Chapitre 1. En particulier nous avons vu comment la stratégie basée sur une marche aléatoire fournit une solution très simple qui ne demande pas de prétraitement, n'étant néanmoins pas optimale.

Le but de chapitre section est de présenter d'autres méthodes, également très classiques, qui permettent de résoudre ce problème avec de meilleures performances, et notamment de répondre aux requêtes en temps $O(\log n)$ sans demander plus de $O(n)$ espace mémoire.

Notez qu'en classe nous verrons encore une autre approche, beaucoup plus récente et très efficace en grandes dimensions : le *Locality-Sensitive Hashing* (LSH) [IM98]. C'est cette approche qui nous a décidés à placer ce chapitre en deuxième partie de poly, en tant que sujet de recherche ayant connu des avancées majeures récentes.

5.1.1 Méthode du Slab

La première méthode présentée dans cette section n'est pas encore optimale (en termes de taille de la structure et temps de prétraitement), mais a l'avantage d'être simple à analyser et rapide à mettre en place.

Comme le suggère le nom (*slab* en anglais) la stratégie qu'on suit consiste à partitionner le plan en bandes verticales, chacune définie par des paires de droites passant par les sommets de la subdivision. Comme la subdivision a taille n, il existe au plus $O(n)$ sommets (extrémités des segments de la subdivision) ainsi que $\Theta(n)$ bandes verticales (car aucun point ne reste dans l'intérieur d'une bande). La structure de données stoque, pour chaque bande b, la liste $E_b(\mathcal{G})$ des arêtes de la subdivision qui intersèctent une bande donnée b : ces arêtes sont stoquées dans un tableau, triées par ordre croissant des ordonnées (qui est bien défini, car deux arêtes n'ont pas d'extrémité commune à l'intérieure d'une même bande).

La procédure pour trouver le voisin plus proche d'un point donné q est assez naturelle : on effectue d'abord une recherche binaire pour déterminer la bande verticale contenant q, et ensuite il reste à faire une autre recherche binaire pour trouver le segment qui est au-dessus de q. Étant donné que chaque bande contient au plus $O(n)$ segment, cette procédure est assez efficace et nécessite au plus temps $O(\log n)$. Pour la même raison, l'espace mémoire peut atteindre une taille de l'ordre $\Theta(n)$, lorsque on a $\Theta(n)$ bandes verticales, chacune contenant $\Theta(n)$ segments. Ce qui est exprimé par le théorème suivant.

Théorème 5.1 (Dobkin et Lipton [DL76])**.** *Étant donné une subdivision planaire $\mathcal{G}$ de taille n, il existe une structure de données de taille $O(n^2)$ (utilisant $O(n^2)$ mots machine) qui permet de*

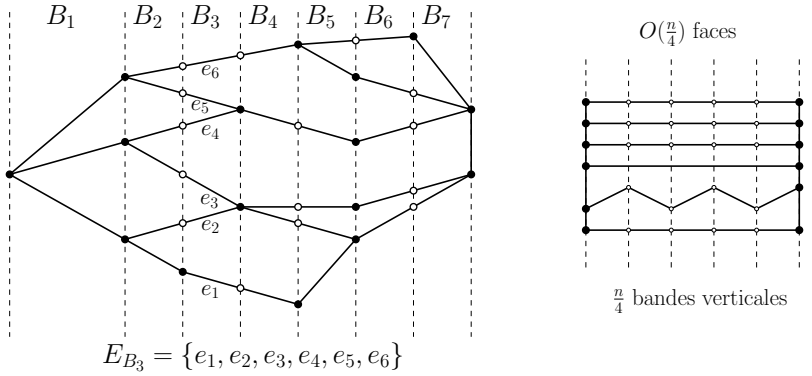


FIGURE 5.1 – *Localisation dans une subdivision planaire avec la méthode du* Slab *: pour chaque bande verticale on stoque les arêtes dans l'ordre des ordonnées croissant. Sur la gauche une subdivision planaire avec la correspondante décomposition en bandes verticales : les cercles noirs représentent les sommets originaux de la décomposition. A droite est présenté un exemple où la complexité de la subdivision en bandes verticales est quadratique.*

résoudre le planar point location problem *en temps optimal* $O(\log n)$ *(pire cas). La construction d'une telle structure demande temps* $O(n^2 \log n)$ *dans le pire cas.*

Amélioration à base de persistance Toujours en utilisant une stratégie de balayage et une décomposition trapézoidale du plan, Sarnak et Lipton [ST86] ont montré comment réduire la taille de la structure à $O(n)$ mots machine, ainsi que le temps nécessaire pour son calcul ($O(n \log n)$). Avec quelques simplifications, on peut obtenir une structure similaire, comme suggéré par l'exercice suivant.

Exercice 5.2 (Persistent search trees)**.** *Prouver que la méthode précédente, basée sur une décomposition en bandes verticales, peut être améliorée en réduisant la taille de l'espace utilisé à* $O(n \log n)$ *mots machines et la complexité du prétraitement à un temps* $O(n \log n)$*.*

5.1.2 Triangulations hiérarchiques

La solution proposée par Kirkpatrick [Kir83] est basée sur une décomposition hiérarchique d'une triangulation planaire. On suppose que la subdivision dont on dispose est une triangulation, où toutes les faces, y comprises la face externe, sont triangulaires. Si ce n'est pas le cas, on peut toujours se ramener à ce cas-ci en triangulant toutes les faces de la subdivision de départ. Les propriétés montrées dans la partie restante de cette section permettent d'établir le résultat suivant :

Théorème 5.3. *Étant donnée une triangulation planaire* $\mathcal{T}$ *de taille* n*, il existe une structure de données permettant d'effectuer la localisation planaire en temps optimal* $O(\log n)$*. Une telle structure requiert espace* $O(n)$ *et peut se construire en temps* $O(n)$*.*

Description de la structure L'idée principale repose sur une représentation hiérarchique : on part de la triangulation initiale $\mathcal{T}$ et on construit une suite de k triangulations $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_k$ satisfaisant les conditions suivantes :

— $\mathcal{T}_0 = \mathcal{T}$ et $\mathcal{T}_k$ ne contient qu'un seul triangle,
— $k = O(\log n)$,

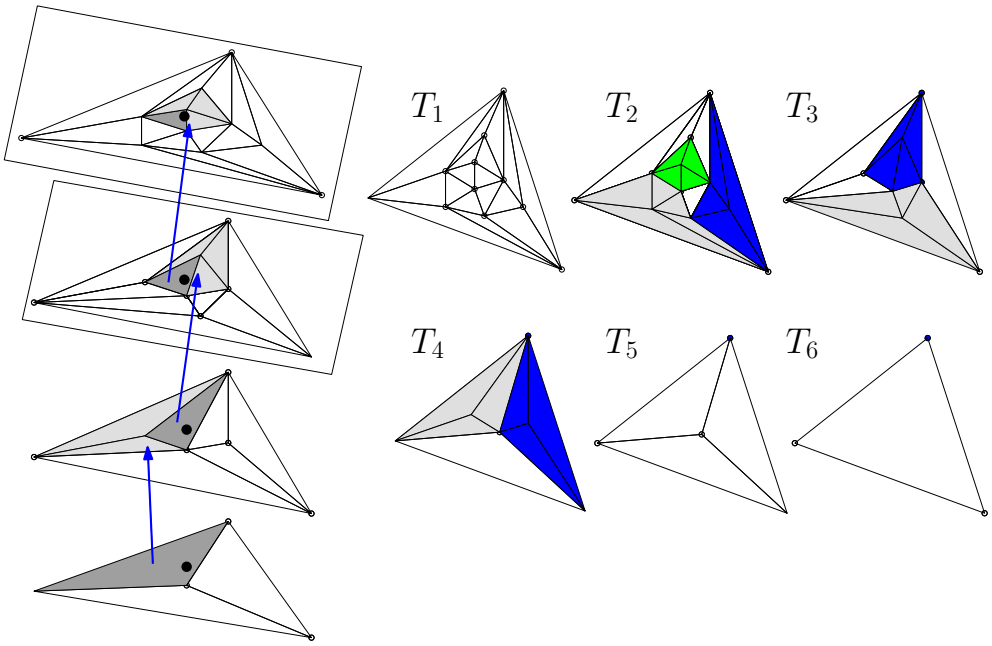


Figure 5.2 – *La structure de données hiérarchique conçue par Kirkpatrick, permettant la localisation d'un point dans une triangulation en temps* $O(\log n)$.

— T_{i+1} s'obtient à partir de T_i en supprimant un ensemble de sommets indépendants et en triangulant les faces correspondantes.

La validité de cette approche repose sur le lemme suivant, dont la preuve fait intervenir la formule d'Euler.

Lemme 5.4. *Étant donné une triangulation planaire* $\mathcal{T}$ *ayant* n *sommets, il existe un ensemble indépendant constitué d'au moins* $\frac{n}{18}$ *sommets, chacun de degré au plus* 8*. Un tel ensemble peut se calculer en temps* $O(n)$*.*

Démonstration. Comme $\mathcal{T}$ est une triangulation planaire, l'une des conséquences de la formule d'Euler est que le nombre d'arêtes est $e = 3n - 6$ et le degré moyen des sommets est 6 :

$$\sum_v degré(v) = 2e = 6n - 12$$

On va prouver que le nombre de sommets ayant degré au plus 8 est au moins $\frac{n}{2}$. Raisonnons par l'absurde, en supposant qu'il existe plus de $\frac{n}{2}$ sommets de degré au moins 9 (notons par V_9 l'ensemble de sommets de $\mathcal{T}$ ayant degré au moins 9). Dans ce cas il en résulte que le degré moyen des sommets devient trop grand, car

$$\sum_{v\in V} degré(v) = \sum_{v\in V_9} 9 + \sum_{v\in V\setminus V_9} degré(v) > 9\cdot\frac{n}{2} + 3\cdot\frac{n}{2} = 6n > 6n - 12$$

(rappelons que dans une triangulation planaire le degré des sommets est au moins 3).

Pour la construction de l'ensemble indépendant on peut procéder de manière greedy. A chaque étape il suffit de choisir un sommet v_j de degré au plus 8 et de supprimer l'étoile de triangles autour de v. Le prochain sommet v_{j+1} sera choisi parmi les sommets (de degré au plus 8) qui ne sont pas adjacents aux sommets déjà supprimés. Ainsi, à chaque étape il faut écarter

au plus 9 sommets : le sommet v_j plus ses voisins (au plus 8, par construction). Ces remarques garantissent que l'ensemble est indépendant, et sa taille est $\geq (\frac{n}{2})/9 = \frac{n}{18}$.

□

Analyse des performances

Lemme 5.5. *Pour une triangulation planaire de taille n la structure hiérarchique consiste de $O(\log n)$ niveaux, et peut se construire en temps $O(n)$, utilisant espace $O(n)$.*

Démonstration. Pour le lemme précédent, on sait que T_{i+1} a été construite à partir de T_i en supprimant au moins $O(n)$ sommets internes. Ainsi on a $|T_i| \leq C|T_i|$ (pour une certaine constante C) et $T_{k-1} = 1$, ce qui conduit à $|T_0| \leq C^k$: et puisque on a $|T_0| = n$, on obtient $k = O(\log n)$. Concernant le temps de calcul de la structure, il suffit d'observer que chaque sommet de T_i est supprimé exactement une fois, et le coût de la mise à jour (suppression plus retriangulation des faces) est $O(d)$ pour un sommet de degré d : à chaque suppression de sommet on enlève d arêtes et on en rajoute $d-3$, ce qui se fait en temps $O(|T_i|)$ pour la triangulation à niveau i. Puisque la taille de l'ensemble indépendant est au moins $\frac{n}{18}$ sommets, le coût global de mise à jour de toutes les triangulations intermédiaires T_i est donné par :

$$\sum_{i \in O(\log n)} O(|T_i|) \leq \sum_{i \in O(\log n)} Cn(\frac{17}{18})^i = O(n)$$

Un argument similaire permet d'établir que la structure nécessite espace $O(n)$. □

Lemme 5.6. *Avec la méthode hiérarchique la localisation d'un point p dans une triangulation planaire de taille n se fait en temps $O(\log n)$.*

Démonstration. La localisation commence par localiser p dans la triangulation T_{k-1} : puisque T_k possède un seul triangle (contenant le point p par définition), il suffit de tester l'un des trois triangle de T_{k-1} pour localiser p. De manière similaire, on peut retrouver le triangle contenant p dans la triangulation T_i à partir de l'information donnée par la triangulation T_{i+1} (où par induction on a déjà localisé p) : cette étape ne demande qu-un temps $O(1)$ puisque, par construction, T_{i+1} a été obtenue en supprimant un sommet de degré borné, ce qui permet de tester un nombre borné de triangles pour localiser p. Et finalement, il ne reste qu'à observer que ce procédé sera répété au plus $O(\log n)$ fois, à cause du nombre de niveaux de la structure hiérarchique. □

5.2 Range searching et recherche du plus proche voisin

Nous allons étudier une classe de problèmes, très étudiés, connus sous le nom de *range search*. Dans cette section nous présenterons un aperçu de quelques résultats célèbres dans ce domaine. De manière générale le but est de concevoir des structures de données permettant de répondre efficacement aux requêtes, tout en gardant au minimum les ressources mémoires utilisées : bien sur, on se donne le droit de faire un prétraitement des données en entrée.

Par exemple, dans le cas de l'*orthogonal range search*, on a :

— une entrée constituée d'un ensemble $\mathcal{S} = \{p_1, \ldots, p_n\}$ de points dans $\mathbb{R}^d$,

— une requête, définie par un d-rectangle $R = [x_1, y_1] \times \ldots [x_d, y_d] \subset \mathbb{R}^d$,

— le résultat de la requête correspond à $\mathcal{S} \bigcap R$ (les points contenus dans la région R). Parfois on se limite à demander le nombre $k = |\mathcal{S} \bigcap R|$.

Problème 5.7 ((Orthogonal) range queries)**.** *Étant donné un ensemble $\mathcal{S} = \{p_1, \ldots, p_n\}$ de points dans $\mathbb{R}^d$ et un rectangle $R = [x_1, y_1] \times \ldots [x_d, y_d] \subset \mathbb{R}^d$ déterminer les points contenus dans R (ou leur nombre).*

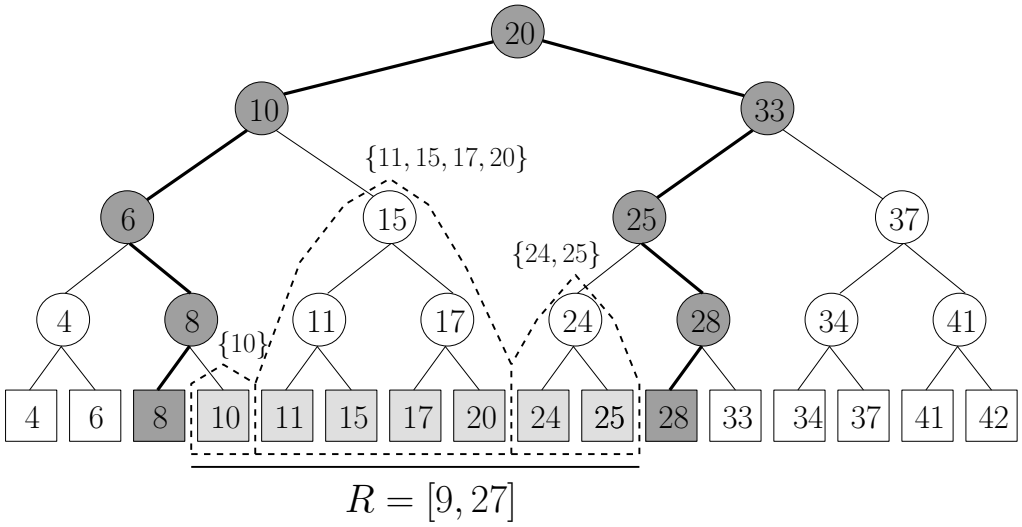


Figure 5.3 – *Range queries en dimension* 1 *: la solution est basée sur l'utilisation d'arbres binaires de recherche équilibrés.*

Exercice : range search en $1D$**.** En dimension 1, on dispose de n nombres réels, et il faut déterminer quels (ou combien de) valeurs se trouvent dans une intervalle donnée $[a, b]$. Ce problème admet une solution assez simple, utilisant une structure de données bien connue : les *arbres binaires de recherche équilibrés*. Le temps que nécessite le prétraitement est clairement $O(n \log n)$, alors que l'espace utilisé est $O(n)$.

Question Comment peut-on répondre à une requête en temps $O(\log n + k)$ (où k est le nombre de points dans $[a, b]$) ?

5.2.1 Range trees

La solution qu'on va esquisser s'inspire de l'exemple donné auparavant pour le cas 1-dimensionnel. Le but est d'établir la validité du résultat suivant :

Théorème 5.8. *Étant donnés* n *points* $\mathcal{S} = \{p_1, \ldots, p_n\}$ *dans* $\mathbb{R}^2$ *et un rectangle* $R = ([a_1, b_1] \times [a_2, b_2])$*, il existe une structure de données permettant de lister les points contenus dans* R *en temps* $O(k + \log^2 n)$*. Une telle structure requiert espace* $O(n \log n)$ *et peut se construire en temps* $O(n \log n)$*.*

La stratégie à suivre consiste concevoir une structure à deux niveaux, en utilisant encore des arbres binaires de recherche équilibrés. On va construire un arbre T, stoquant dans ses nœuds les abscisses x_i de points dans $\mathcal{S}$ (les points sont ordonnés par abscisses croissantes) de la manière suivante :

- à chaque nœud $v \in T$ on associe un *ensemble canonique* C_v constitué des points stoqués dans le sous-arbre enraciné en v.
- pour chaque nœud v on construit un arbre binaire de recherche équilibré T_v pour stoquer l'ensemble canonique C_v : cette fois, les clés stoquées dans les nœuds sont les ordonnées des points dans C_v.

On a ainsi décrit une structure de données arborescente à deux niveaux, qui peut se voir globalement comme un *arbre d'arbres*. Le premier niveau consiste d'un x-range tree, dont chaque nœud est associé à un y-range tree.

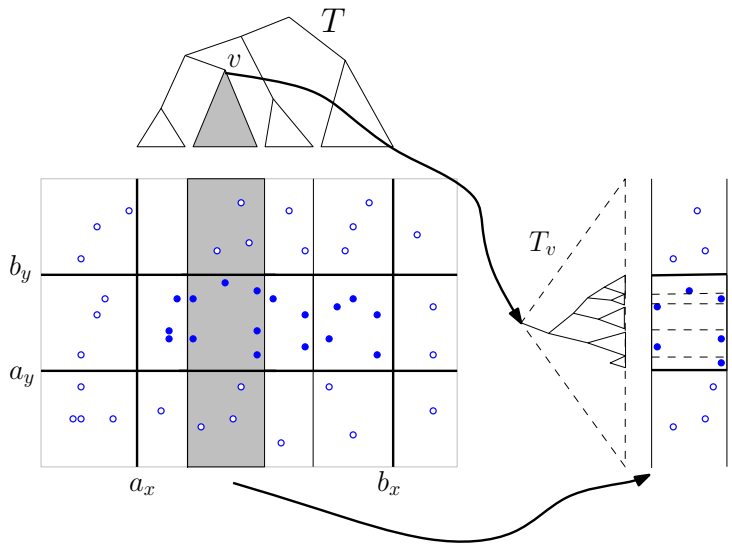


FIGURE 5.4 – *Un exemple d'utilisation des Range Trees.*

Pour répondre aux requêtes on peut procéder de la manière suivante. La première étape consiste à déterminer quels sont les nœuds de l'arbre T dont les ensembles canoniques contiennent des points p tels que $a_1 \leq p_x \leq b_1$. Observons qu-ils existe au plus $O(\log n)$ sommets v de T satisfaisant cette contrainte. Pour la manière dont on a construit le range tree, tous les points p qui satisfont $a_1 \leq p_x \leq b_1$ appartiennent à l'union disjointe des ensembles $\bigcup_v C_v$, où C_v est l'ensemble canonique associé au nœud v. Il reste maintenant à interroger la structure par rapport à la coordonnée y. Cela peut se faire à l'aide de l'arbre T_v, pour chaque nœud v.

Les performances des *range trees* décrites pas le théorème 5.8 dérivent de la validité des lemmes suivants :

Lemme 5.9. *Étant donné n points dans $\mathbb{R}^2$, et une requête orthogonale $[a_1, b_1] \times [a_2, b_2]$, on peut interroger un* range tree *en temps $O(k + \log^2 n)$.*

Démonstration. La première étape, celle où on détermine les nœuds associés aux ensembles canoniques, prend temps $O(\log n)$, car il s'agit d'une recherche en dimension 1, comme déjà décrit auparavant. Pour tout nœud v de T trouvé ainsi, il faut à nouveau interroger un range tree T_v : comme il y a $O(\log n)$ arbres à interroger, chacun demandant temps $O(\log n)$, la complexité totale peut s'évaluer à $O(\log^2 n)$. Pour chaque arbre T_v, on peut lister les k_v points à retourner en temps $O(k_v)$: et comme $k = \sum_v k_v$, le temps total pour lister tous les points est $O(k)$. □

Lemme 5.10. *La construction d'un* range tree *peut se faire en temps $O(n \log n)$ et nécessite espace $O(n \log n)$.*

Démonstration. Compte tenu de la taille de l'arbre T, le stoquage du premier niveau nécessite espace $O(n)$. Il reste maintenant à évaluer l'espace utilisé pour stoquer tous les arbres constituant le deuxième niveau. Étant donné un ensemble canonique C_v contenant $|C_v|$ points, le range tree T_v correspondant possède $|C_v|$ feuilles, et donc $O(|C_v|)$ nœuds au total, puisque il est équilibré. Mais il est plus convenable d'évaluer le nombre d'arbres auxquels un point $p \in \mathcal{S}$ peut appartenir. Observons qu'un point p dans C_v, appartient à T_v ainsi qu'à tous les autres arbres T_a, où le nœud a est ancêtre de v dans T. Puisque l'arbre T est équilibré, chaque sommet possède au plus $O(\log n)$ ancêtres, et donc chaque sommet appartient à au plus $O(\log n)$ arbres. Ainsi, puisque

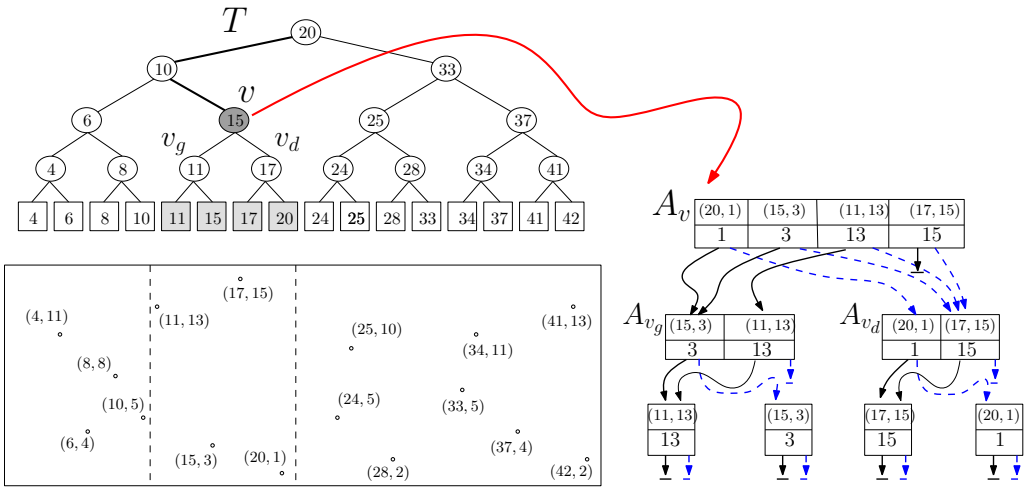


FIGURE 5.5 – *Cet exemple illustre la technique connue sous le nom de* Fractional Cascading.

chaque point apparaît globalement $O(\log n)$ fois, le nombre de feuilles (et donc de nœuds) de tous les arbres T_v est au total $O(n \log n)$, ce qui conclut la preuve. □

5.2.1.1 Orthogonal Range Search en dimension supérieure

On va maintenant esquisser la stratégie à suivre pour généraliser les range trees décrits ci-dessus au cas de points en dimension $d > 2$: on supposera toujours que $d = O(1)$.

L'idée est d'appliquer le schéma précédent récursivement : une recherche en dimension d est récursivement réduite à une recherche en dimension 1.

On commence par créer un range tree T par rapport à la première coordonnée x_1. Pour tout ensemble canonique de T, on construit un range tree de dimension $d-1$, où l'on tient compte des coordonnées restantes $(x_2, \ldots, x_d)$.

Il est alors clair comment effectuer une requête en dimension d. On détermine d'abord les ensembles canoniques associés à l'intervalle $[a_1, b_1]$. Et ensuite on procède récursivement, en interrogeant chaque ensemble canonique, avec la requête $[a_2, b_2] \times \ldots \times [a_d, b_d]$.

Théorème 5.11. *Étant donnés n points $\mathcal{S} = \{p_1, \ldots, p_n\}$ dans $\mathbb{R}^d$ les* range trees *permettent de lister les points contenus dans une région orthogonale en temps $O(k + \log^d n)$. En dimension d, les* range trees *nécessitent espace $O(n \log^{d-1} n)$ et peuvent se construire en temps $O(n \log^{d-1} n)$.*

5.2.1.2 Amélioration : Fractional Cascading

Le temps de recherche peut être améliorer à l'aide d'une technique, connue sous le nom de *Fractional Cascading*, introduite dans un cadre algorithmique plus général par Chazelle et Guibas [CG86]. Il est possible d'exploiter une propriété assez simple mettant en relation les arbres associés au sommets de l'arbre v : effet, si un sommet v a deux fils u et w alors $T_v = T_u \cup T_w$. Cette simple remarque, combinée avec une stratégie sophistiquée pour gérer la fusion des arbres T_v, permet de baisser le temps d'une requête à $O(\log n)$ en dimension 2.

Théorème 5.12. *Étant donnés n points en $\mathbb{R}^d$ il est possible de répondre à des requêtes dans une région orthogonale en temps $O(k + \log^{d-1} n)$. La structure de données nécessite un espace $O(n \log^{d-1} n)$ et peut se construire en temps $O(n \log^{d-1} n)$.*

La solution repose sur une stratégie assez astucieuse, dont nous allons esquisser les grandes lignes. Pour ce faire, rappelons la raisons pour laquelle, en dimension 2, les range trees fournissent un temps de requête $O(\log^2 n)$. La première étape consiste à localiser, dans l'arbre T les nœuds v correspondant à l'intervalle de requête (sur l'axe des abscisses) : cette étape prend naturellement un temps $O(\log n)$. Dans un deuxième temps, pour tout nœud v ainsi localisé dans T, il faut effectuer une recherche dans l'arbre correspondant T_v, dont les éléments correspondent aux ordonnées des points (stoqués par ordre croissant). Et comme cette étape demande aussi un temps $O(\log n)$ dans le cas le pire, on atteint finalement une complexité $O(\log^2 n)$.

La stratégie à mettre en place consiste à enrichir la structure de données avec un certain nombre d'informations (des pointeurs) afin de réduire le temps de requête dans les arbres T_v : le but est de stoquer une quantité d'information suffisante de manière à pouvoir localiser un sommet dans T_v en temps $O(1)$. Comme illustré par la Fig. 5.5, on va construire un graphe ayant une "structure arborescente" dont les sommets correspondent aux nœuds de l'arbre T. Pour chaque sommet v dans T on va stoquer un tableau A_v contenant les éléments de l'ensemble canonique correspondant à v, triés selon les valeurs des y (par ordre croissant). Pour chaque élément $i \in A_v$ on stoque deux pointeurs vers les sommets correspondants respectivement aux nœuds v_g et v_d (les racines des sous-arbres gauche et droit de v dans T). Ces deux références pointent respectivement vers l'élément plus grand ou égal dans les deux tableaux A_{v_g} et A_{v_d} correspondant à v_g et v_d : observons que A_v s'obtient comme union disjointe des tableaux A_{v_g} et A_{v_d}. Remarquons d'abord que si l'on connaît, lors de la visite de T, la position d'un élément dans un tableau A_v, alors on connaît aussi la position de cet élément dans le sous-tableau au niveau suivant, et cela en temps $O(1)$ grâce aux références stoquées. Ces informations permettent de localiser et temps $O(1)$ un sommet dans T_v, lorsqu'on effectue la visite de l'arbre T pour localiser v, comme l'on va montrer ci-dessous.

La recherche commence de la racine de T, comme pour le cas des range trees : dans une première phase on va localiser les $O(\log n)$ nœuds de T correspondant aux ensembles canoniques qui contiennent les points (les abscisses) de la région $R = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$ (requête). A une certaine étape on tombe sur un sommet $v_{split} \in T$ d'où partent deux chemins distincts : à l'aide d'une recherche binaire, on peut trouver en temps $O(\log n)$ le plus petit élément dans $A_{v_{split}}$ avec ordonnées $e_y \geq y_{min}$ (et de manière similaire pour y_{max}). Ce procédé est répété tout au long de la visite de l'arbre T, jusqu'à ce que trouve tous les $O(\log n)$ nœuds $v_0, v_1, \ldots,$ qui correspondent aux ensembles canoniques souhaités $C_0, C_1, \ldots$.

Pour terminer, il reste à déterminer quels sont les points, dans l'union des C_i, dont les ordonnées sont comprises entre les valeurs y_{min} et y_{max}. Considérons un ensemble canonique C_v correspondant à un sommet $v \in T$ il suffit de se souvenir du point $p_{min} \in C_v$, telle que son ordonnée a la plus petite valeur supérieure ou égale à y_{min} (on fait la même chose pour le point p_{max} défini de manière similaire). Observons qu'on connaît exactement la position du point p_{min} (et de p_{max}) dans le tableau auxiliaire A_v, grâce aux informations repérées lors de la descente dans l'arbre T. Finalement, comme le tableau A_v est trié, il suffit de lister tous les éléments dont les ordonnées se trouvent entre celles de p_{min} et p_{max}.

Pour conclure, observons que la technique qu'on vient d'esquisser permet de gagner un facteur $O(\log n)$ en évitant d'effectuer une recherche binaire dans le dernier niveau de la structure. Ainsi, en dimension d, il restera à effectuer $(d-1)$ recherches (chacune nécessitant un temps au plus $O(\log n)$).

5.2.2 kd-trees : plus proche voisin

Le but de cette section est de présenter une autre structure de données, appelée *kd-trees*, qui permet de résoudre efficacement certains problèmes de recherche géométriques. Entre autre, les *kd*-trees, fournissent une solution efficace pour la recherche du plus proche voisin en petite

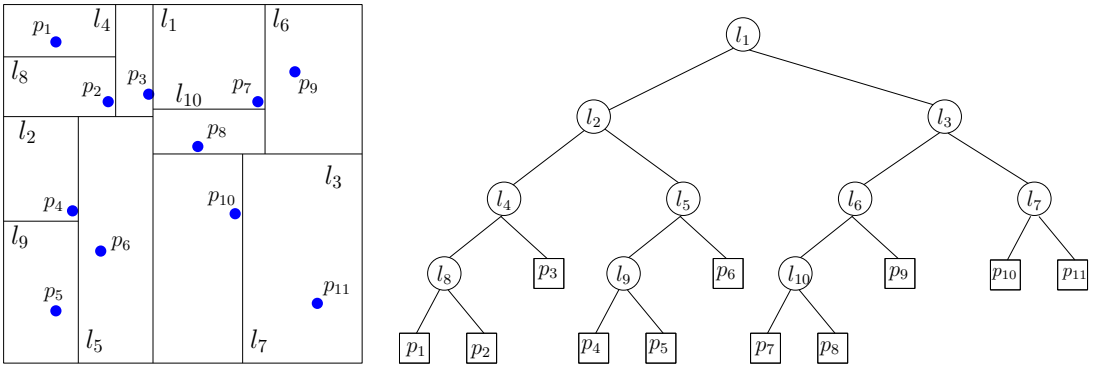


Figure 5.6 – *Un exemple de partitionnement de l'espace utilisant les kd-trees (pour $k = 2$).*

dimension : comme déjà mentionné au Chapitre 1, ce problème consiste à trouver le point p appartenant à $\mathcal{S} = \{p_1, \ldots, p_n\} \subset \mathbb{R}^k$, qui est le plus proche de q (le point requête, qui n'appartient pas forcement à $\mathcal{S}$).

La stratégie principale à la base des *kd*-trees est la construction d'une décomposition de l'espace en hyperrectangles. Par rapport à d'autres stratégies de décomposition (par exemple utilisant des grilles régulières, ou des *quad-trees*), celle basée sur les *kd*-trees a plusieurs avantages, à niveau du temps de requête ainsi que de ressources mémoire utilisées, comme on le verra dans la suite.

Description de la structure On définit une procédure récursive de l'espace, utilisant des hyperplans séparateurs. A l'étape i, on choisit un hyperplan parallèle aux axes qui sépare en deux ensembles de "même" taille les points appartenant a un hyperrectangle existant à l'étape $i - 1$.

Cette décomposition de l'espace est décrite par une structure de données arborescente, formée par un arbre binaire dont les nœuds correspondent aux hyperrectangle. Chaque nœud de l'arbre stoque plusieurs informations : la direction orthogonale à l'hyperplan de coupe, ainsi que sa position ; et bien sur les références vers le fils gauche et droit. Si à chaque étape on réussit à séparer les points en sous-ensembles de taille équivalentes, l'arbre résultant sera équilibré : pour n points on aura ainsi $O(\log n)$ niveaux, ce qui garantira un temps de requête efficace.

Un point crucial concerne le choix de l'hyperplan séparant, et comme l'on peut imaginer il existe plusieurs manières de partitionner un ensemble de points en deux parties ayant à peu près la même taille. On peut par exemple déterminer la médiane par rapport à la direction de coupe. Ou encore, on peut couper avec un hyperplan dont la position est donnée par le barycentre des points à séparer.

Dans le premier cas on s'assure que l'arbre sera équilibré : mais cela peut produire des hyperrectangles allongés pour certaines distributions de points (clairement non-uniformes), ce qui s'avère mauvais dans la recherche du plus proche voisin. Dans le second cas, le choix du barycentre permet d'obtenir des régions plus "carrées", mais l'arbre résultant pourrait ne pas être équilibré.

Recherche du plus proche voisin La procédure de recherche consiste en un parcours récursif de l'arbre à partir de la racine. Cette recherche est guidée à chaque étape par l'une des coordonnées du point q (le point requête) : l'algorithme visite le sous-arbre gauche (ou droit) selon que la coordonnée q_{x_i} est plus petite (ou grande) que la coordonnée correspondante stoquée dans le nœud courant de l'arbre (celle qui donne la position de l'hyperplan séparateur).

L'algorithme ne termine pas lorsqu'une feuille est atteinte : dans ce cas on sait seulement que le point p stoqué dans cette feuille est un candidat possible parmi les voisins de q, le plus proche voisin pouvant se trouver dans une cellule géométriquement proche de celle de p (on marque p comme le meilleur candidat). Ainsi, une deuxième phase se rend nécessaire pour remonter dans l'arbre et mettre à jour le meilleur candidat éventuellement. La recherche d'un nouveau meilleur candidat fait intervenir aussi les points se trouvant de l'autre coté par rapport à l'hyperplan séparateur. Cette phase nécessite de calculer l'intersection de l'hyperplan séparateur avec une hypersphère centrée en q : cela fait intervenir un certain nombre de détails techniques (concernant les cas d'intersection possibles entre hyperplan et hypersphère) dont il faut tenir compte avec soin.

Performances des *kd*-trees

Lemme 5.13. *Un kd-tree permet de déterminer le plus proche voisin d'un ensemble de n points en temps $O(\log n)$ lorsque les points sont repartis uniformément. La construction peut se faire en temps $O(n \log^2 n)$, et l'espace mémoire est linéaire.*

Observons que le temps mémoire peut être réduit à $O(n \log n)$ si pour le calcul de la médiane à chaque étape on utilise un algorithme de complexité $O(n)$, à la place du simple tri des points qui demande $O(n \log n)$.

Concernant le temps de requête, on a déjà observé que la borne $O(\log n)$ est valable pour des ensembles de points bien distribués. De manière plus sophistiquées on peut montrer que dans le pire des cas le temps de calcul est au plus $O(kN^{1-\frac{1}{k}})$, où N est le nombre de nœuds de la structure : ce qui est mieux qu'une recherche linéaire lorsque la dimension k est petite par rapport à N.

Et pour conclure cette digression sur les *kd*-trees on peut mentionner qu'ils fournissent une solution différente au problème de l'orthogonal range search. Le temps de requête dans ce cas devient $O(n^{1-\frac{1}{k}})$.

Bibliographie

[CG86] Bernard Chazelle and Leonidas J. Guibas. Fractional cascading : I. a data structuring technique. *Algorithmica*, 1(2) :133–162, 1986.

[DL76] David P. Dobkin and Richard J. Lipton. Multidimensional searching problems. *SIAM J. Comput.*, 5(2) :181–186, 1976.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors : towards removing the curse of dimensionality. In *Proc. 30th ACM Sympos. on Theory of Computing*, pages 604–613, 1998.

[Kir83] David G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1) :28–35, 1983.

[ST86] Neil Sarnak and Robert Endre Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29(7) :669–679, 1986.

Chapitre 6

Algorithmes d'approximation géométrique

6.1 Quelques mots sur la complexité des algorithmes

Nous renvoyons le lecteur au cours INF561 sur les algorithmes et la complexité pour une introduction formelle de la théorie de la complexité. Ici nous nous contenterons d'en donner un aperçu informel afin de rendre la suite du chapitre plus accessible au néophyte.

Modèles de calcul. Du point de vue de l'algorithmique, la difficulté d'un problème se mesure par le temps de calcul et l'espace mémoire nécessaires à un ordinateur moderne pour résoudre n'importe quelle instance du problème. Pour les besoins de l'analyse l'ordinateur est représenté par un modèle de calcul abstrait approprié, qui la plupart du temps est le modèle *real-RAM* : grosso modo, la machine est munie d'un processeur séquentiel capable d'effectuer chaque opération arithmétique de base (addition, soustraction, multiplication, division) en temps constant, disons en un cycle de calcul, et qui a accès à n'importe quelle case de la mémoire également en temps constant, chaque case pouvant stocker un nombre réel avec une précision arbitraire. La complexité est alors mesurée en nombre de cycles du processeur nécessaires pour la résolution du problème sur n'importe quelle entrée. D'autres modèles existent, comme par exemple le modèle de comparaison, où seules les comparaisons entre nombres effectuées par le processeur comptent (c'est le modèle utilisé par exemple pour l'analyse des algorithmes de tri).

Classes de complexité. Une fois le modèle fixé, les problèmes sont répartis en différentes catégories (appelées *classes de complexité*) selon leur difficulté. Ces catégories peuvent changer d'un modèle à l'autre, toutefois elles sont choisies suffisamment larges pour être communes à plusieurs modèles. Par exemple, les classes P et NP, définies initialement pour le modèle des machines de Turing, restent identiques pour le modèle real-RAM. Pour faire simple, la classe P est celle des problèmes dont la solution peut être découverte en temps polynômial en la taille de l'entrée, tandis que la classe NP est celle des problèmes dont la solution peut être vérifiée en temps polynômial. Clairement, P est incluse dans NP, mais il n'est pas clair que les deux classes soient égales (c'est la fameuse conjecture *P=NP*, qui demeure ouverte à ce jour). Par exemple, étant donné une expression booléenne de longueur polynômiale en n, où n est le nombre de variables booléennes $x_1, \cdots, x_n$ mises en jeu, le problème SAT consiste à trouver une assignation de valeurs $\{x_1, \cdots, x_n\} \to \{0, 1\}$ telle que l'expression devienne vraie. Vérifier si une assignation de valeurs donnée rend l'expression vraie se fait aisément en temps polynômial. Par contre, il n'est pas clair qu'une telle assignation puisse être découverte en temps polynômial.

NP-complétude. P et NP sont les deux classes de complexité qui nous intéressent ici. Les problèmes dans NP ne sont pas tous de la même difficulté, et en fait NP se subdivise en toute une hiérarchie de problèmes. Parmi ceux-ci, il est intéressant de savoir lesquels sont *les plus difficiles*. La notion théorique d'être parmi les problèmes les plus difficiles de NP est appelée *NP-complétude*. Elle se définit très élégamment comme suit : un problème $p \in NP$ est NP-complet si tout autre problème p' de NP peut être réduit au problème p par une réduction polynômiale. En d'autres termes, p est NP-complet si sa résolution permet de résoudre tous les problèmes de NP, modulo une phase d'adaptation des données et des résultats qui ne prend qu'un temps polynômial. Notez qu'il n'est pas exclu que la résolution de p elle-même requiert un temps exponentiel. Parmi les problèmes NP-complets, SAT est certainement le plus connu. Une manière standard de prouver qu'un problème p donné est NP-complet consiste à réduire un autre problème connu comme étant NP-complet (comme par exemple SAT) à p par une réduction polynômiale. Il est intéressant de noter que l'égalité P=NP, si elle est vraie, pourrait être démontrée par le même mécanisme : il suffirait pour cela de trouver une solution polynômiale à un probleme NP-complet arbitrarire, comme par exemple SAT. Il est donc particulièrement frustrant de constater que cette question n'a toujours pas été tranchée à l'heure actuelle.

Problèmes d'optimisation et algorithmes d'approximation. Une fois que l'on sait qu'un problème p donné est NP-complet, que fait-on ? Si P=NP, alors cette information n'a pas grand intérêt puisque finalement toutes les instances de p peuvent être résolues en temps polynômial. Par contre, si P est différent de NP, alors certaines instances ne peuvent être résolues en temps polynômial et nécessiteront des temps de calculs arbitrairement longs en pratique. Dès lors, il faut trouver une solution de repli, par exemple en réduisant un peu nos exigences. C'est ce que fait la théorie algorithmique de l'approximation, qui s'adresse à une catégorie particulière de problèmes appelés problèmes d'optimisation. Ces problèmes ont pour objet de calculer des structures qui maximisent ou minimisent une fonction de coût donnée. Le problème de l'arbre couvrant minimal en est un exemple : étant donné un graphe pondéré G (c'est l'entrée du problème), calculer un arbre couvrant de G dont le poids est minimal. Ici, le poids de l'arbre (i.e. la longueur totale des arêtes de l'arbre) est la fonction de coût. Le choix de la fonction de coût influence radicalement la nature et la complexité du problème : alors que certaines fonctions de coût peuvent rendre le problème NP-complet, d'autres peuvent le ramener dans P. C'est le cas par exemple du problème appelé Max-Cut, dont l'objectif, étant donné un graphe pondéré G, est de séparer l'ensemble des sommets de G en deux sous-ensembles disjoints V_1, V_2 de manière à ce que le poids total des arêtes reliant V_1 à V_2 soit le plus grand possible. Ce problème est bien connu pour être NP-complet. Mais si on change la fonction de coût en prenant l'opposé de la somme des poids des arêtes du cut, alors le problème devient celui de trouver le cut dont le poids total initial des arêtes est minimal. Ce nouveau problème, appelé Min-Cut, est dans P.

Formellement, étant donné un problème d'optimisation (disons de minimisation) p associé à une fonction de coût $|\cdot|$, pour toute instance i de p on note $\text{OPT}_p(i)$ une solution optimale du problème p vis-à-vis de la fonction de coût $|\cdot|$ sur cette instance. En d'autres termes, toute solution s à p sur l'instance i vérifie $|s| \geq |\text{OPT}_p(i)|$. Étant donné un paramètre $\kappa \geq 1$, un algorithme de κ-approximation de p est un algorithme A qui prend en entrée n'importe quelle instance i de p et retourne en temps polynômial une solution $A(i)$ telle que $|\text{OPT}_p(i)| \leq |A(i)| \leq \kappa|\text{OPT}_p(i)|$. Comme on le verra, certains problèmes ne sont pas approximables, même pour des facteurs κ dépendant fortement de la taille de l'entrée. D'autres problèmes en revanche peuvent être approximés avec une précision arbitrairement bonne. Formellement, pour un tel problème p il existe une famille d'algorithmes polynômiaux $\{A_\varepsilon\}_{\varepsilon>0}$ telle que pour tout $\varepsilon > 0$ l'algorithme A_ε approxime la solution de p avec une précision $1 + \varepsilon$, autrement dit : pour tout

$\varepsilon > 0$ et toute instance i du problème, on a $|\text{OPT}_p(i)| \leq |A_\varepsilon(i)| \leq (1+\varepsilon)|\text{OPT}_p(i)|$. Une telle famille d'algorithmes $\{A_\varepsilon\}_{\varepsilon>0}$ est appelée *schéma polynômial d'approximation*, ou *polynomial-time approximation scheme* (PTAS) en anglais. L'acronyme PTAS sera utilisé dans la suite pour alléger le propos.

Note sur le contenu du chapitre. Dans la section 6.2 nous illustrons les concepts introduits ci-dessus au travers d'un exemple de problème d'optimisation particulier connu sous le nom de problème du voyageur de commerce. Ce dernier a un grand intérêt pédagogique puisque, bien qu'il ne soit pas approximable en temps polynômial, il admet des variantes de nature très géométrique qui le sont (certaines admettent même des PTAS). Il couvre donc bien les différents cas de figure présentés dans cette introduction, en plus de faire appel à des outils de géométrie algorithmique pour sa résolution. En classe nous verrons également une autre application des algorithmes d'approximation : la réduction de la complexité de certains problèmes dans P. Bien que ces problèmes soient déjà résolubles en temps polynômial, le degré des polynômes régissant leur complexité peut être élevé, rendant ainsi les méthodes inefficaces voire impossibles à mettre en œuvre en pratique. Calculer des solutions approchées peut permettre de réduire considérablement la complexité des problèmes et de les rendre ainsi résolubles efficacement en pratique, comme nous le verrons au travers d'un exemple particulier : celui du calcul de médians généralisés en toutes dimensions. Pour information, bien que ce problème soit un classique de la géométrie algorithmique, il demeure en grande partie ouvert et suscite encore un fort intérêt de la part de la communauté aujourd'hui. D'ailleurs, l'algorithme qui sera présenté en classe est très récent puisqu'il date de 2009 [MS09].

6.2 Le problème du voyageur de commerce

Un cycle hamiltonien dans un graphe est un cycle qui passe exactement une fois par chaque sommet. Étant donné un graphe simple G, le problème Hamiltonian Cycle consiste à déterminer s'il existe un cycle hamiltonien dans G, et éventuellement à exhiber un tel cycle dans l'affirmative.

Théorème 6.1. *Hamiltonian Cycle est NP-complet.*

Le problème du voyageur de commerce est en quelque sorte la version d'optimisation de Hamiltonian Cycle. Connu sous le nom de Travelling Salesman Problem (TSP) en anglais, il consiste, étant donné un graphe complet pondéré G à poids positifs, à trouver un cycle hamiltonien dont les arêtes ont un poids total minimal. Notez qu'il existe toujours un tel cycle puisque G est complet, donc le problème ne porte plus sur l'existence d'un cycle hamiltonien, mais plutôt sur son coût.

Théorème 6.2. *TSP est NP-complet. De plus, si P est différent de NP, alors pour toute fonction calculable en temps polynômial $\alpha : \mathbb{N} \to \mathbb{R}^+$, TSP ne peut être approximé en temps polynômial avec un facteur d'approximation de $1 + \alpha(n)$, où n est la taille de l'entrée.*

Notez que le facteur d'approximation a le droit de dépendre ou non de la taille n de l'entrée. Malgré cela, il n'existe aucun algorithme capable de l'atteindre en temps polynômial.

Démonstration. Le cas particulier où la fonction α est identiquement nulle revient à dire que TSP est NP-difficile (et donc NP-complet puisque TSP est dans NP). Il nous suffit donc de démontrer la deuxième partie de l'énoncé du théorème. Supposons qu'il existe un algorithme polynômial A qui $(1+\alpha(n))$-approxime TSP. Nous allons alors opérer une réduction polynômiale depuis Hamiltonian Cycle, ce qui nous permettra de résoudre ce dernier en temps polynômial et ainsi d'obtenir une contradiction avec le théorème 6.1.

Soit $G = (V, E)$ un graphe simple, non pondéré, qui peut être complet ou non. Nous construisons à partir de G un nouveau graphe $G' = (V', E')$, pondéré et complet cette fois, de la manière suivante :

- $V' = V$,
- pour toute arête e dans E, ajouter l'arête pondérée $(e, 1)$ dans E',
- pour toute arête e qui n'est pas dans E, ajouter l'arête pondérée $(e, n(1 + \alpha(n)))$ dans E', où n est la taille de V.

Dans G', tout cycle hamiltonien ne contenant que des arêtes de E a un poids total de n, tandis que tout autre cycle hamiltonien a un poids total strictement supérieur à $n(1 + \alpha(n))$. Dès lors, deux cas de figure se présentent à nous :

— Si G contient un cycle hamiltonien, alors ce cycle a un poids total de n dans G'. L'algorithme A va alors retourner un cycle de poids total au plus $n(1 + \alpha(n))$, puisqu'il $(1 + \alpha(n))$-approxime TSP. Mais comme tous les cycles qui ne sont pas entièrement contenus dans le sous-graphe G ont un poids total strictement supérieur à $n(1 + \alpha(n))$, l'algorithme va forcément retourner un cycle hamiltonien de G.

— Si G ne contient aucun cycle hamiltonien, alors l'algorithme va forcément retourner un cycle qui n'est pas entièrement contenu dans le sous-graphe G.

Au final, on a un critère simple pour décider si G a un cycle hamiltonien : il suffit de vérifier si la sortie de l'algorithme A est entièrement contenue dans G. Ainsi, on a réduit le problème Hamiltonian Cycle à l'approximation de TSP avec un facteur d'approximation de $1 + \alpha(n)$. Comme α est calculable en temps polynômial, la constructon du graphe pondéré G' à partir de G s'effectue en temps polynômial. Quant à la vérification de la solution fournie par l'algorithme A, elle se fait en temps polynômial également. Ainsi, nous pouvons résoudre Hamiltonian Cycle en temps polynômial, ce qui contredit le théorème 6.1. □

6.2.1 Metric-TSP

Une variante plus géométrique de TSP consiste à considérer des graphes métriques, i.e. des graphes complets pondérés dont les poids (positifs) vérifient l'inégalité triangulaire : pour tous sommets u, v, w, $|uw| \leq |uv| + |vw|$. Cette variante est appelée Metric-TSP.

Théorème 6.3. *Il existe un algorithme polynômial qui approxime Metric-TSP avec un facteur d'approximation de* $\frac{3}{2}$.

Démonstration. L'algorithme en question a été introduit par Christofides [Chr76]. On va d'abord présenter une version simplifiée de l'algorithme qui fournit une 2-approximation. Puis on décrira les modifications à apporter pour obtenir une $\frac{3}{2}$-approximation. Étant donné un graphe métrique G, on procède en quatre étapes :

1. on construit un arbre couvrant minimal T de G,
2. on double chaque arête de l'arbre T afin d'en faire un (multi-)graphe connexe G_T dont tous les sommets ont degré pair,
3. on calcule un cycle eulérien dans G_T,
4. on transforme le cycle eulérien de G_T en cycle hamiltonien de G en court-circuitant les sommets déjà visités.

La procédure est illustrée dans les quatre premières images de la figure 6.1. Son implémentation requiert quelques explications complémentaires. La construction de l'arbre couvrant minimal T à l'étape 1 se fait en temps polynômial par l'algorithme de Kruskal. Le graphe G_T obtenu à la fin de l'étape 2 est dit eulérien car il est connexe et tous ses sommets ont degré pair. Il admet donc des cycles eulériens, i.e. des cycles passant une et une seule fois par chaque arête de G_T. La construction d'un cycle eulérien à l'étape 3 se fait en temps polynômial par l'algorithme de Fleury : on part d'un sommet quelconque de G_T, puis on construit le cycle pas à pas, en

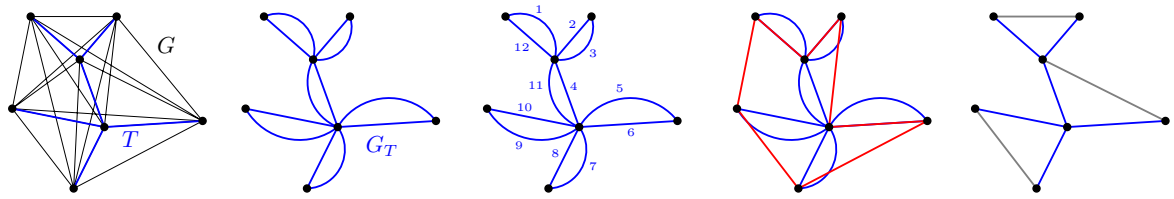


Figure 6.1 – *Illustration de l'algorithme de Christofides. De gauche à droite : l'arbre couvrant minimal T, le graphe G_T obtenu par dédoublement des arêtes de T, le cycle eulérien calculé dans G_T, et enfin le cycle hamiltonien dans G obtenu à partir du cycle eulérien en court-circuitant certaines arêtes. La dernière image à droite montre le graphe G_T utilisé dans la version plus élaborée de l'algorithme, obtenu par ajout dans l'arbre T des arêtes d'un matching parfait de poids minimal (en gris).*

traversant à chaque pas une arête dont la suppression ne déconnecte pas le graphe (sauf à laisser des sommets isolés) ; une fois traversée, l'arête est supprimée du graphe. À la fin du processus il ne peut rester aucune arête dans le graphe car on a bien fait attention de ne jamais le deconnecter ; de plus, pour la même raison on doit être retourné au point de départ. Enfin, l'étape 4 se fait simplement en parcourant pas à pas le cycle calculé à l'étape 3, et à chaque pas en se rendant directement au prochain sommet de G non encore visité (cette opération de court-circuit est rendue possible par le fait que G est complet). On obtient ainsi un cycle hamiltonien c de G.

Quel est le poids total du cycle c comparé à celui d'un cycle optimal OPT ? Deux remarques vont nous aider : d'une part, si on supprime une arête de OPT alors on obtient un arbre couvrant de G, de poids total au moins égal à celui de T. Ainsi, on a $|\text{OPT}| \geq |T|$. D'autre part, le cycle eulérien construit à l'étape 3 passe exactement 2 fois par chaque arête de T, donc son poids total est $2|T|$. Or, à l'étape 4 on ne fait que court-circuiter ce cycle, et donc par l'inégalité triangulaire le cycle c obtenu est forcément de poids total inférieur ou égal. On conclut que $|c| \leq 2|T| \leq 2|\text{OPT}|$, ce qui signifie que notre procédure construit des 2-approximations de Metric-TSP.

L'algorithme de Christofides est une variante de notre procédure dans laquelle l'étape 2 est remplacée par l'étape 2bis ci-dessous (voir l'image de droite dans la figure 6.1 pour une illustration) :

2bis. Soit W l'ensemble des sommets de l'arbre T qui ont degré impair. Comme le degré total de l'arbre est pair, le nombre d'éléments dans W est également pair. À partir du sous-graphe complet de G engendré par W, on construit un matching parfait de poids total minimal, i.e. un ensemble d'arêtes deux-à-deux disjointes qui apparient les points de W et dont le poids total est minimal. On ajoute ces arêtes à T et on appelle G_T le graphe ainsi obtenu.

Le calcul du matching parfait de poids minimal se fait en temps polynômial en utilisant par exemple l'algorithme hongrois. Comparé à T, le nouveau graphe G_T obtenu contient une arête de plus par sommet de T de degré impair. Il est donc eulérien et permet ainsi aux étapes 3 et 4 ci-dessus d'être exécutées. Pour borner supérieurement le poids du cycle hamiltonien c obtenu, considérons un cycle hamiltonien optimal OPT_W sur le sous-graphe de G engendré par W. En supprimant une arête sur deux dans OPT_W on obtient un matching parfait ; en échangeant les arêtes du cycle supprimées et celles gardées on obtient un autre matching parfait. Commes les ensembles d'arêtes impliquées dans les deux matchings sont disjoints, l'un au moins de ces deux matchings a un poids total inférieur ou égal à $\frac{1}{2}|\text{OPT}_W|$, et donc le matching parfait de poids minimal calculé à l'étape 2bis a un poids d'au plus $\frac{1}{2}|\text{OPT}_W|$, qui par l'inégalité triangulaire est borné supérieurement par $\frac{1}{2}|\text{OPT}|$. Ainsi, on a $|G_T| \leq |T| + \frac{1}{2}|\text{OPT}| \leq \frac{3}{2}|\text{OPT}|$. Cette relation est

meilleure que celle obtenue avec la version basique de l'algorithme ($|G_T| \leq 2|\text{OPT}|$). Le reste de l'analyse reste inchangé et donne un facteur d'approximation de $\frac{3}{2}$. □

Le problème Metric-TSP est connu pour n'admettre aucun PTAS. En particulier, on sait qu'il n'existe pas d'algorithme d'approximation pour des facteurs d'approximation plus petits que $\frac{220}{219} \approx 1.00456$ [PV00]. Le meilleur facteur d'approximation atteint pour le moment est précisément celui de l'algorithme de Christofides, c'est-à-dire $\frac{3}{2}$. Aucun progrès n'a été fait de ce côté-là depuis son introduction, excepté pour certains cas particuliers comme par exemple lorsque les poids des arêtes valent 1 ou 2. Actuellement la conjecture la plus communément admise est que Metric-TSP admet des algorithmes d'approximation pour des facteurs d'approximation descendant jusqu'à $\frac{4}{3}$.

6.2.2 Euclidean-TSP

Une variante plus spécialisée de Metric-TSP considère le graphe des distances euclidiennes entre les points d'un nuage dans $\mathbb{R}^d$. Cette variante est appelée Euclidean-TSP. Que nous apporte cette restriction ? En quelques mots, l'espace euclidien $\mathbb{R}^d$ n'est pas un espace métrique quelconque : c'est un espace affine dans lequel la métrique est issue d'un produit scalaire. Et comme nous allons le voir, cela fait toute la différence !

Théorème 6.4. *Euclidean-TSP admet un PTAS.*

Ce résultat dû à S. Arora [Aro98] a fait date car il montre qu'un problème a priori aussi difficile que TSP peut en fin de compte admettre des solutions raisonnables à condition de trouver les bonnes variantes du problème et d'avoir recours à l'approximation. Le reste de la section 6.2.2 est dédié entièrement à la preuve du théorème 6.4, qui suit en gros celle d'Arora. Pour simplifier nous nous concentrerons sur le cas planaire ($\mathbb{R}^2$), mais l'approche est générale et marche en toutes dimensions, modulo un facteur polynômial supplémentaire dans le temps de calcul [1].

L'approche en bref. Nous allons mettre au point un algorithme qui fournit une $(1 + O(\varepsilon))$-approximation de (planar) Euclidean-TSP en temps polynômial. La constante cachée dans le grand-O est absolue et ne dépend pas des données, donc elle peut aisément être ramenée à 1 par une adaptation des paramètres dans chacune des étapes de l'algorithme [2]. Étant donné un nuage de points P dans le plan, l'algorithme procède selon les étapes suivantes :

1. Il renormalise P par une homothétie, puis il projette les points de P sur leur plus proche sommet sur une grille régulière de pas unité.
2. Il subdivise récursivement le plus petit carré contenant P avec un quadtree aligné sur les lignes de la grille.
3. Il place des points particuliers sur la grille, appelés *portails*.
4. Il calcule le plus court cycle polygonal dans $\mathbb{R}^2$ qui passe par tous les sommets de la grille sur lesquels les points de P ont été projetés, et qui ne traverse les arêtes de la grille qu'au niveau des portails.
5. Il parcourt le cycle polygonal calculé à l'étape 4 et court-circuite chaque sommet du cycle qui n'est pas sommet de la grille. Puis il modifie le cycle obtenu pour le faire passer par les points de P au lieu des sommets de la grille, afin d'obtenir un cycle hamiltonien dans le graphe complet reliant les points de P.

1. Notez que le degré du polynôme dépend de la dimension, qui doit donc être fixée et considérée comme une constante.

2. Nous choisissons de ne pas adapter les paramètres en ce sens dans notre description afin de rendre leur choix plus intuitif pour le lecteur.

La procédure est illustrée dans la figure 6.2. Nous allons maintenant détailler les étapes ci-dessus une à une, en montrant pour chacune comment elle influence la longueur du chemin obtenu. Les différents résultats seront combinés dans la sous-section 6.2.2.6.

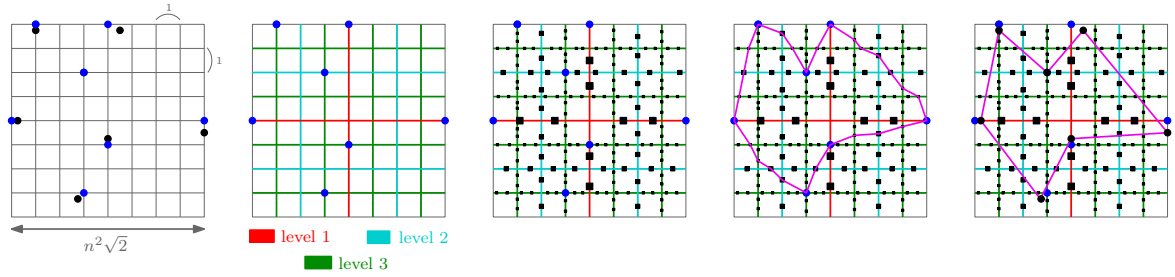


Figure 6.2 – *Les étapes de l'algorithme. De gauche à droite : renormalisation et projection du nuage P sur une grille unité, construction du quadtree, mise en place des portails, calcul du plus court cycle polygonal respectant les portails, et enfin conversion du cycle polygonal en cycle hamiltonien reliant les points de P.*

6.2.2.1 Renormalisation et projection

Soit n le nombre de points de P. On renormalise P par une homothétie d'un facteur s de manière à obtenir un nuage P_s dont le plus petit carré englobant aligné sur les axes mesure $n^2\sqrt{2}$ de côté. L'image de tout cycle hamiltonien c sur P par l'homothétie de rapport s est un cycle hamiltonien de P_s de longueur $s|c|$. Dès lors, l'image d'un cycle hamiltonien minimal sur P par l'homothétie de rapport s est un cycle hamiltonien minimal sur P_s, et de même l'image de toute $(1+\varepsilon)$-approximation sur P est une $(1+\varepsilon)$-approximation sur P_s, et réciproquement. Nous pouvons donc considérer à partir de maintenant, et sans perte de généralité, que le nuage de points P est déjà renormalisé.

Considérons maintenant une grille régulière de pas unité alignée avec les axes. Le choix de la grille est unique modulo une translation dans le carré unité $[0,1)^2$, et nous verrons dans la sous-section 6.2.2.6 qu'il est judicieux de fixer la translation de manière aléatoire. Une fois la grille unité choisie, on projette chaque point $p \in P$ sur le sommet de la grille le plus proche. Si jamais p a plusieurs plus proches sommets, alors on projette p sur l'un d'eux, choisi arbitrairement. Appelons g la fonction de projection ainsi définie. Elle n'est pas forcément injective puisque plusieurs points de P peuvent être projetés sur un même sommet de la grille. Comme g déplace chaque point de P d'au plus $\sqrt{2}/2$, l'image par g de tout cycle c sur P vérifie $|c| - n\sqrt{2} \leq |g(c)| \leq |c| + n\sqrt{2}$. En appelant respectivement $\textsc{opt}$ et $\textsc{opt}_g$ des cycles hamiltoniens minimaux sur P et $g(P)$, on a donc :

$$|\textsc{opt}_g| \leq |g(\textsc{opt})| \leq |\textsc{opt}| + n\sqrt{2} \leq \left(1 + \frac{1}{2n}\right)|\textsc{opt}|, \tag{6.1}$$

où la dernière inégalité provient du fait que la longueur de tout cycle hamiltonien sur P est au moins $2n^2\sqrt{2}$ puisque le plus petit carré englobant P a des côtés de longueur $n^2\sqrt{2}$. Cette relation nous sera utile dans la section 6.2.2.6 pour relier la longueur de l'approximation calculée sur $g(P)$ à la longueur de $\textsc{opt}$.

6.2.2.2 Construction du quadtree

Soit $k \in \mathbb{N}$ tel que $2^{k-1} \leq n^2\sqrt{2} < 2^k$. On agrandit le plus petit carré englobant P vers la droite et vers le haut, de manière à obtenir un carré C de côté 2^k. Les arêtes et les sommets de C sont alors arêtes et sommets de la grille unité.

On subdivise C en quatre en le coupant au milieu dans chaque dimension, puis on répète l'opération récursivement sur chacune des quatre régions carrées obtenues. On s'arrête lorsque la région à subdiviser est un carré unité (c'est en fait une cellule de la grille). On obtient ainsi une subdivision de C appelée quadtree, qui peut se représenter en mémoire par un arbre 4-aire Q où chaque noeud représente une région particulière à un niveau particulier de la subdivision. En l'occurence, les feuilles de l'arbre correspondent aux cellules de la grille. Voir la figure 6.3 pour une illustration.

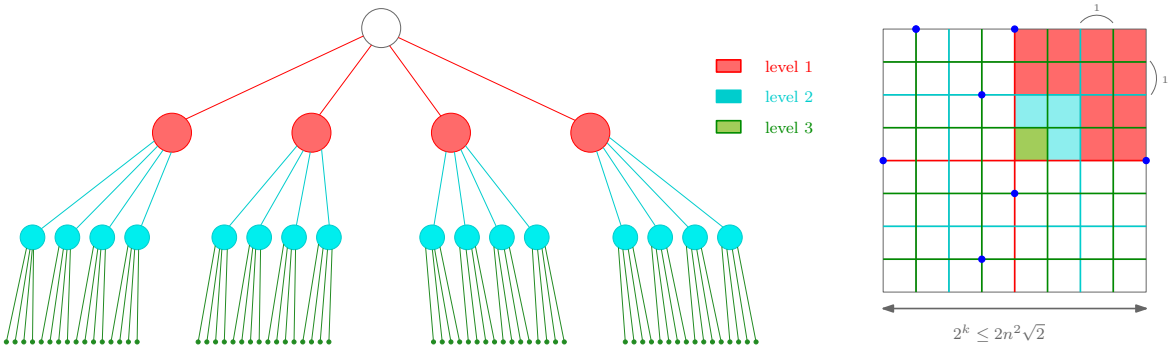


FIGURE 6.3 – *Représentations du quadtree construit à l'étape 2 de l'algorithme.*

Comme le carré C mesure $2^k \leq 2n^2\sqrt{2}$ de côté, le nombre de cellules de la grille situées à l'intérieur de C (et donc le nombre de feuilles dans le quadtree Q) est $O(n^4)$. Ainsi,

Lemme 6.5. *La taille (en nombre de noeuds) du quadtree Q est $O(n^4)$, et sa hauteur est $O(\log n)$.*

6.2.2.3 Mise en place des portails

On place des portails sur les arêtes de la grille, par lesquels pourra passer le cycle polygonal calculé à l'étape 4. Le placement se fait de la manière suivante. Soit $m = \lfloor \frac{\log n}{\varepsilon} \rfloor$. Considérons une droite utilisée au niveau i pour la subdivision de C opérée à l'étape 2 : cette droite traverse exactement 2^i régions du niveau $i-1$ pour les couper en 2, ce qui fait qu'elle participe aux bords communs de 2^i paires de régions. On place $2^i m$ portails régulièrement espacés le long de la droite, de manière à assurer que chaque paire de regions de niveau i incidente est reliée par m portails. Par exemple, sur la troisième image de la figure 6.2, les droites rouges coupent la région de niveau 0 (i.e. le carré C) en 4 régions de niveau 1, et chaque paire de régions adjacentes est reliée par $m = 2$ portails (de manière similaire, toutes les paires de régions adjacentes de même niveau 2 ou 3 sont reliées par 2 portails). Aux $2^i m$ portails répartis régulièrement le long de la droite de niveau i, on ajoute $2^i + 1$ portails, un à chaque coin de région de niveau i sur la droite. Ceci porte à $m+2$ le nombre de portails joignant deux régions quelconques de niveau i adjacentes. Observez maintenant que le bord de chaque région de niveau i est formé uniquement de droites de niveau entre 1 et i, ce qui fait que le nombre total de portails sur le bord de la région est au plus $4m+4$, soit $m+1$ par côté. Ainsi,

Lemme 6.6. *Les portails sont répartis sur la grille de manière à ce que les régions de même niveau adjacentes dans le quadtree soient connectées par $m+1$ portails, et que le bord de chaque région du quadtree contienne au plus $4m+4$ portails, où $m = \lfloor \frac{\log n}{\varepsilon} \rfloor$.*

6.2.2.4 Calcul du plus court cycle polygonal respectant les portails

Nous dirons qu'un cycle polygonal respecte les portails si ses sommets sont tous des points de $g(P)$ ou des portails, s'il ne traverse les lignes de la grille qu'au niveau de ses sommets qui sont

portails [3], et s'il passe une et une seule fois par chaque point de $g(P)$. Ainsi, nous discrétisons la recherche d'un cycle polygonal minimal en regardant ceux qui respectent les portails. Pour l'instant il y a un nombre infini de tels cycles puisqu'on les autorise à passer un nombre arbitraire de fois par chaque portail. Nous allons donc affiner notre analyse afin de diminuer le nombre de chemins possibles. Soit OPT_p un cycle de longueur minimale parmi tous les cycles polygonaux respectant les portails.

Lemme 6.7. OPT_p *visite chaque portail au plus deux fois.*

Démonstration. Supposons tout d'abord que OPT_p passe par un portail p situé sur une ligne sans traverser cette ligne. En d'autres termes, OPT_p atteint p par une cellule de la grille puis reste dans cette cellule après avoir quitté p. Soient q, s les sommets de OPT_p situés juste avant et juste après p le long du cycle. Comme OPT_p respecte les portails, les points q, p, s sont situés dans la même cellule, et donc il suffit de court-circuiter p en reliant q à s directement pour obtenir un autre cycle polygonal respectant les portails de longueur strictement plus petite, ce qui contredit la définition de OPT_p. Ainsi, OPT_p traverse tous les portails qu'il visite.

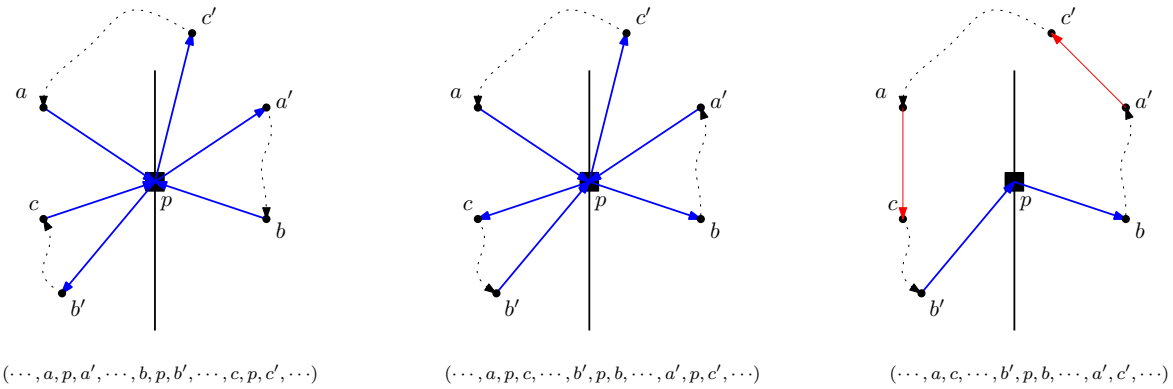


FIGURE 6.4 – *Pour la preuve du lemme 6.7. Le processus de désentrelacement du cycle polygonal au niveau d'un portail p est illustré de gauche à droite.*

Supposons maintenant que OPT_p visite un portail au moins trois fois, tout en le traversant. Soient a, a' les sommets de part et d'autre de p le long de OPT_p lors de la première visite, et soient b, b' et c, c' leurs équivalents lors des autres visites. Le cycle OPT_p est alors de la forme $(\cdots, a, p, a', \cdots, b, p, b', \cdots, c, p, c', \cdots)$. Inversons alors l'orientation de toutes les arêtes de OPT_p entre a' et b et entre b' et c. De plus, inversons le sens des arêtes (c, p), (p, b'), (b, p) et (p, a'). Nous obtenons ainsi un nouveau cycle polygonal $\text{OPT}'_p = (\cdots, a, p, c, \cdots, b', p, b, \cdots, a', p, c', \cdots)$ dont la longueur est la même que celle de OPT_p puisqu'ils ont la même réalisation géométrique dans le plan. Maintenant, on observe que OPT'_p passe deux fois par le portail p sans le traverser : comme plus haut, il suffit donc de relier a à c et a' à c' directement sans passer par p pour obtenir un cycle polygonal strictement plus court, contredisant ainsi la définition de OPT_p. L'argument est illustré dans la figure 6.4. □

Ainsi, grâce au lemme 6.7 l'espace dans lequel on doit chercher OPT_p est fini. Plus précisément, OPT_p peut être défini dans chaque cellule de la grille séparément comme un ensemble de segments reliant les portails de son bord, puis obtenu par concaténation des segments de différentes cellules. Comme d'après le lemme 6.6 le bord de chaque cellule contient $O(m)$ portails, et que

3. Notez que les points de $g(P)$ peuvent être à n'importe quels sommets de la grille, y-compris à ceux qui ne sont pas des portails. Respecter les portails implique que lorsque le cycle atteint un point de $g(P)$ qui n'est pas un portail, il doit ensuite rester dans la même cellule jusqu'à trouver un portail.

chaque portail est sollicité au plus 2 fois, il y a un nombre fini de combinaisons possibles par cellule. Et comme il y a $O(n^4)$ cellules, on en conclut que le nombre total de combinaisons possibles pour la structure de OPT_p est fini. Toutefois, ce nombre est beaucoup trop grand pour une recherche exaustive. Voyons pourquoi :

- À l'intérieur d'une cellule on a $\Theta(m)$ portails avec 3 possibilités pour chacun : 0, 1 ou 2 visites, ce qui donne $3^{\Theta(m)} = n^{\Theta(1/\varepsilon)}$ possibilités au total... ce qui reste raisonnable.
- En revanche, il faut aussi définir les appariements entre portails : or il y en a $\Theta(m!) = n^{\Theta(1/\varepsilon(\log\log n + \log 1/\varepsilon))}$ au total, car ils correspondent grosso modo aux bijections de l'ensemble des portails dans lui-même.

Ainsi, l'espace des configurations pour une seule cellule a une taille de $n^{\Theta(1/\varepsilon(1+\log\log n+\log 1/\varepsilon))}$, ce qui est super-polynômial. Pour réduire ce nombre, nous allons exploiter l'observation suivante :

Lemme 6.8. OPT_p *ne s'auto-intersecte pas, sauf au niveau des portails.*

Démonstration. Notons tout d'abord que par définition OPT_p passe une et une seule fois par chaque point de $g(P)$, donc il ne peut s'auto-intersecter au niveau de ces points. Supposons que OPT_p s'auto-intersecte au niveau d'un point p qui n'est ni un point de $g(P)$ ni un portail. Alors l'intersection se fait au niveau des intérieurs relatifs de deux segments de OPT_p, disons $[a, a']$ et $[b, b']$. Ces segments sont inclus dans la même cellule que p puisque OPT_p respecte les portails. Il suffit donc de connecter a à c et a' à c', puis d'inverser le sens des arêtes de OPT_p entre c et a', pour obtenir un autre cycle polygonal, de description $(\cdots, a, c, \cdots, a', c', \cdots)$, dont la longueur est $|\text{OPT}_p| - \|a - a'\| - \|c - c'\| + \|a - c\| + \|a' - c'\|$, ce qui est strictement plus petit que $|\text{OPT}_p|$ puisque la somme des longueurs des diagonales du quadrilatère convexe (a, c, a', c') est supérieure à la somme des longueurs de deux côtés opposés quelconques. Ainsi, nous obtenons un cycle polygonal de longueur strictement inférieure à celle de OPT_p, ce qui contredit la définition de OPT_p. □

Le lemme 6.8 garantit qu'au sein de chaque cellule de la grille seuls les appariements évitant les intersections des intérieurs relatifs des segments sont valides. Cette contrainte introduit un ordre implicite sur les portails le long du bord de la cellule, qui indentifie chaque appariement valide avec une expression bien parenthésée. Ainsi, le nombre d'appariements valides possibles dans chaque cellule est de l'ordre du nombre d'expressions bien parenthésées avec m paires de parenthèse, c'est-à-dire $O(2^m) = n^{O(1/\varepsilon)}$. Ceci réduit le nombre total de combinaisons à regarder dans chaque cellule à $n^{O(1/\varepsilon)}$, une quantité polynômiale puisque le paramètre d'approximation ε est considéré comme une constante.

Concaténation des ensembles de segments des différentes cellules. Nous savons maintenant comment procéder dans chaque cellule pour explorer toutes les configurations possibles. Mais comment faire à un niveau plus global pour choisir la configuration à garder dans chaque cellule et pour concaténer les configurations des différentes cellules entre elles ? Une recherche exaustive pour chaque cellule ne serait pas raisonnable car $n^{O(1/\varepsilon)}$ combinaisons par cellule pour $O(n^4)$ cellules donnerait un espace à explorer de taille exponentielle $n^{O(n^4/\varepsilon)}$. En fait, toutes les configurations dans les cellules ne donnent pas naissance à un cycle (ou même à un chemin) après concaténation. De plus, nous recherchons le cycle de longueur minimale, donc nous pouvons rapidement éliminer certaines configurations plus coûteuses que d'autres localement en terme de longueur. L'idée est de procéder par **programmation dynamique** sur le quadtree, afin d'exploiter ces indices locaux et d'éliminer ainsi une grande partie des configurations. On parcourt le quadtree de bas en haut, en partant des feuilles et en terminant à la racine. La table de requêtes, représentée dans la figure 6.5, est le produit du quadtree par les ensembles de configurations dans les cellules. D'après le lemme 6.5 et ce que nous venons de dire plus haut,

la taille de la table de requêtes est $O(n^{4+O(1/\varepsilon)})$ et donc polynômiale. La procédure se déroule comme suit :

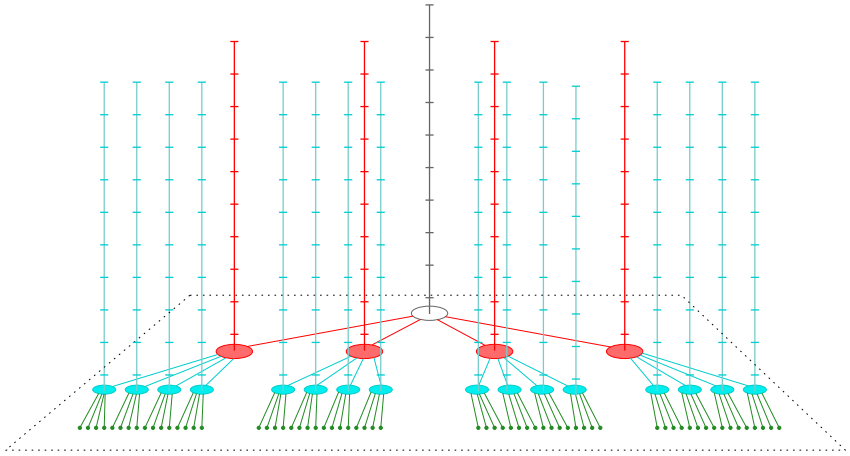


Figure 6.5 – *La table de requêtes. Le quadtree de la figure 6.3 est représenté dans le plan horizontal, duquel partent des segments verticaux représentant les différentes sélections possibles de portails au niveau de chaque nœud.*

- On commence par les feuilles, qui comme on l'a vu correspondent aux cellules de la grille. Pour chacune d'entre elles on regarde chaque sélection possible de portails sur son bord (rappelez vous que chaque portail peut être utilisé 0, 1 ou 2 fois, d'après le lemme 6.7), et pour chaque sélection on calcule l'appariement entre portails (et entre portails et points de $g(P)$ situés dans la cellule) qui donne l'ensemble de segments de longueur minimale. Cet appariement est ensuite stocké dans la table. Comme il y a $n^{O(1/\varepsilon)}$ sélections de portails et pour chacune d'elle $n^{O(1/\varepsilon)}$ choix possibles d'appariements, le coût total du traitement d'une feuille est $n^{O(1/\varepsilon)}$.
- On remonte ensuite dans le quadtree, inspectant tous les nœuds d'un même niveau avant de passer aux niveaux supérieurs. Chaque nœud correspond à une région R carrée du plan, dont le bord contient $O(m) = O(\log n/\varepsilon)$ portails. On regarde toutes les sélections possibles parmi ces $O(m)$ portails, et pour chaque sélection on regarde toutes les sélections de portails possibles pour les 4 fils du nœud qui sont compatibles entre elles et avec la sélection faite pour le nœud lui-même. Par une recherche exaustive, cela nous fait $n^{O(1/\varepsilon)}$ configurations possibles à regarder, chacune d'elles nécessitant un temps $O(m)$ pour vérifier que les configurations sur les bords des régions des fils sont compatibles entre elles et avec celle de R. Pour chacune d'elles, on récupère dans la table de requêtes les chemins polygonaux les plus courts au sein des 4 fils, et on les concatène via les portails. On stocke dans la table de requêtes le chemin polygonal le plus court construit de cette manière pour la sélection de portails particulière choisie sur le bord de la région R. En fait, on n'a même pas besoin de concaténer les chemins à ce stade : il suffit de stocker dans la table 4 pointeurs et un entier : les 4 pointeurs indiquent que pour la sélection de portails considérée sur le bord de la région R, le chemin le plus court est obtenu en choisissant une sélection de portails particulière sur chaque fils du nœud, et la longueur du chemin obtenu (stockée dans l'entier) est simplement la somme des longueurs des sous-chemins dans les 4 fils. Ainsi, l'opération de stockage coûte un temps constant. Au total, le traitement d'un noeud avec toutes les sélections de portails possibles sur le bord de sa région et sur celles de ses fils coûte un temps $n^{O(1/\varepsilon)}O(m) = n^{O(1/\varepsilon)}O(\log n/\varepsilon)$.

Le processus de programmation dynamique s'achève lorsque la région R à traiter est le carré

englobant C, qui correspond à la racine du quadtree. On retourne alors, parmi toutes les configurations de portails des fils de la racine qui sont compatibles entre elles, celle qui donne le cycle polygonal de longueur minimale. Pour construire ce cycle, il suffit alors de parcourir le quadtree récursivement : à la descente dans l'arbre on récupère à chaque couple (nœud, sélection de portails) les 4 couples correspondants sur les 4 fils du nœud ; au niveau des feuilles on récupère les segments donnés par l'appariement optimal entre portails et entre portails et points de $g(\varepsilon)$; enfin, à la remontée dans l'arbre on concatène au niveau de chaque noeud les chemins polygonaux récupérés de ses fils. Cette opération de concaténation nous coûte un temps linéaire en la longueur totale du chemin, qui est au plus $O(n^4 \log n/\varepsilon)$ puisqu'il y a $O(n^4)$ feuilles et chacune apporte un nombre de segments au plus proportionnel au nombre de ses portails, soit $O(m) = O(\log n/\varepsilon)$. Comme lors du parcours récursif du quadtree seule une sélection de portails est considérée par nœud, le nombre total de concaténations est au plus le nombre de nœud du quadtree, soit $O(n^4)$ d'après le lemme 6.5. Ainsi, une fois la programmation dynamique terminée, la construction du cycle polygonal minimal prend un temps polynômial.

Le cycle ainsi obtenu est garanti d'avoir la plus petite longueur possible parmi tous les cycles polygonaux respectant les portails. En effet, pour chaque noeud et chaque sélection de portails sur le bord de sa région R considérés, le chemin polygonal le plus court compatible avec cette sélection est forcément obtenu par concaténation de chemins polygonaux de longueur minimale dans les quatre sous-régions de R. Ceci garantit qu'à chaque étape de la programmation dynamique on calcule, pour le nœud courant et pour la sélection de portails courante, le chemin polygonal compatible le plus court. Ainsi,

Lemme 6.9. *La procédure de programmation dynamique décrite ci-dessus calcule en temps polynômial un cycle polygonal* OPT_p *de longueur minimale parmi ceux qui respectent les portails.*

6.2.2.5 Modification du cycle polygonal en cycle hamiltonien sur P

Transformer OPT_p en cycle hamiltonien sur $g(P)$ est facile : il suffit de court-circuiter les sommets de OPT_p situés entre deux points de $g(P)$ consécutifs, en reliant ces deux points directement. On obtient ainsi un chemin hamiltonien sur $g(P)$, de longueur inférieure ou égale à $|\text{OPT}_p|$. Comment modifier ce chemin pour en faire un chemin hamiltonien sur P ? Rappelons que la projection g n'est pas injective, puisque plusieurs points $p_1, \cdots, p_k$ de P peuvent être projetés sur le même sommet v de la grille. Dans ce cas, comme OPT_p ne passe qu'une seule fois par v, on le fait passer successivement par tous les points p_i. Plus précisément, si OPT_p est décrit localement par la séquence de sommets $\cdots, u, v, w, \cdots$, alors on le transforme en $\cdots, u, p_1, p_2, \cdots, p_k, w, \cdots$. Comme g déplace les points de P d'au plus $\sqrt{2}/2$, le surcoût en terme de longueur engendré par cette modification est d'au plus $k\sqrt{2}$. Dès lors, la longueur totale du chemin hamiltonien c sur P ainsi construit est au plus $|\text{OPT}_p| + n\sqrt{2}$. On a donc

$$|\text{OPT}| \leq |c| \leq |\text{OPT}_p| + n\sqrt{2} \leq |\text{OPT}_p| + \frac{1}{2n}|\text{OPT}|, \tag{6.2}$$

où la dernière inégalité provient du fait que la longueur de tout cycle hamiltonien sur P est au moins $2n^2\sqrt{2}$ puisque le plus petit carré englobant P a des côtés de longueur $n^2\sqrt{2}$, d'après la section 6.2.2.1.

6.2.2.6 Final

Ainsi, les étapes 1 à 5 de la procédure construisent un cycle hamiltonien c sur P de longueur au plus $|\text{OPT}_p| + \frac{1}{2n}|\text{OPT}|$, d'après l'équation 6.2. Reste à borner cette quantité en fonction de $|\text{OPT}|$ seulement, ce qui revient à borner $|\text{OPT}_p|$ en fonction de $|\text{OPT}|$, ou encore (d'après

l'équation 6.1) à borner $|\text{OPT}_p|$ en fonction de $|\text{OPT}_g|$. Malheureusement, comme nous le verrons en classe, il est des cas où le chemin OPT_p a une longueur supérieure à $\kappa|\text{OPT}_g|$, où $\kappa > 1$ est une constante indépendante de ε. Ceci rend impossible la $(1 + O(\varepsilon))$-approximation de $|\text{OPT}_g|$ par $|\text{OPT}_p|$ (et donc celle de $|\text{OPT}|$ par $|c|$) pour des valeurs de ε arbitrairement petites. Toutefois, notre contre-exemple s'appuie sur un positionnement bien particulier de la grille unité par rapport au nuage de points P. Rappelez-vous d'après la section 6.2.2.1 que le positionnement de la grille peut être fait arbitrairement à une translation près dans le carré unité. L'astuce pour éviter les cas pathologiques consiste à choisir une translation aléatoire dans ce carré [4]. On peut alors montrer que l'espérance de la différence entre $|\text{OPT}_p|$ et $|\text{OPT}_g|$ est de l'ordre de ε. On en déduit que, par ce processus de randomisation de la grille, on obtient un cycle hamiltonien c sur P dont la longueur moyenne est $(1 + O(\varepsilon))|\text{OPT}| + \frac{1}{2n}|\text{OPT}|$, c'est-à-dire $(1 + O(\varepsilon))|\text{OPT}|$ pour n suffisamment large (typiquement pour $n \geq 1/\varepsilon$, qui est une constante puisque ε est traité comme tel). Mieux : on peut même dérandomiser le processus en ne considérant qu'un nombre fini (de l'ordre de n^4) de translations t_i de la grille, et pour chacune d'elle en calculant le cycle hamiltonien c_i obtenu par les étapes 1 à 5 décrites plus haut. À la fin du processus on retourne le cycle $c = c_i$ de longueur minimale. On a alors la garantie que $|c| = (1 + O(\varepsilon))|\text{OPT}|$, ce qui était notre objectif. Ainsi, modulo un surcoût polynômial en temps de calcul, nous pouvons construire de manière déterministe un cycle hamiltonien sur P dont la longueur $(1 + O(\varepsilon))$-approxime l'optimal. Ceci conclut la preuve du théorème 6.4. Les détails de la randomisation et de la dérandomisation de la grille seront donnés en classe et peuvent être trouvés dans l'excellent livre de V. Vazirani sur les algorithmes d'approximation [Vaz02]. Pour les détails les plus techniques des preuves, voir l'article original d'Arora [Aro98].

4. En fait, pour des raisons techniques on choisira une translation aléatoire dans le carré englobant C, ce qui permettra de décaler d'autant la subdivision récursive du domaine en quadtree effectuée à l'étape 2.

Bibliographie

[Aro98] S. Arora. Polynomial-time approximation schemes for euclidean tsp and other geometric problems. *Journal of the ACM*, 45(5) :753–782, 1998.

[CEM+96] Kenneth L. Clarkson, David Eppstein, Gary L. Miller, Carl Sturtivant, and Shang-Hua Teng. Approximating center points with iterative radon points. *Int. J. Comput. Geometry Appl.*, 6(3), 1996.

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[MS09] Gary L. Miller and Donald Sheehy. Approximate center points with proofs. In *Symposium on Computational Geometry*, pages 153–158, 2009.

[PV00] Christos H. Papadimitriou and Santosh Vempala. On the approximability of the traveling salesman problem (extended abstract). In *STOC '00 : Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 126–133, 2000.

[Vaz02] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2002. 2nd Edition.

Index

algorithme
  Arora, 100
  balayage, 18
  bascule d'arêtes, 31
  Christofides, 99
  d'approximation, 96
  division-fusion, 20
  Graham, 18
  Guibas-Oudot, 78
  Guibas-Stolfi, 34
  incrémental, 34
  Iterated-Radon, 108
  Iterated-Tverberg, 109
  Jarvis, 18
  Lipton-Tarjan, 60
  marche aléatoire, 11
  Tutte, 49
approximation
  algorithme d', 96
  schéma polynômial d', *voir* PTAS
arbre, 5
  couvrant, 5
    minimal, 32
arithmétique
  d'intervalle, 26
  exacte, 25
  flottante, 24
axe médian, 70

carte, 39
  combinatoire, 40
center point, 16, 107
combinaison
  barycentrique, 13
complexe
  cellulaire, 6
  de témoins, 76
  simplicial, 5
complexité
  classes de, 95
  modèle de calcul, 95
  NP-complétude, 95
convexe
  ensemble, 14
  enveloppe, 13
    borne inférieure, 22
    calcul, 18, 20
coordonnées
  barycentriques, 13
cycle
  periphérique, 52

Delaunay
  angle minimal, 31
  calcul, 34
  espace des sphères, 28
  localement vs globalement, 29
  séquence des angles, 31
  simplexe, 28
  taille, 33
  triangulation, 28
  triangulation restreinte, 73
dessin de graphes
  Koebe, 43
  Méthodes spectrales, 53
  Schnyder, 49, 55
  Tutte, 50, 51
diagramme
  affine, 38
  de Laguerre, 38
  de puissance, 38
  de Voronoï, 29
  de Voronoï restreint, 73

echantillonnage
  ε-échantillon, 70
  échantillon λ-creux, 78
ensemble convexe, *voir* convexe
enveloppe
  affine, 13
  convexe, *voir* convexe
Euler
  formule d', 6

filtrage
  dynamique, 26
  statique, 26

graphe
  definition, 3
  dessin planaire, 3
  plan maximal, 41
  planaire, 4, 41
  planaire 3-connexe, 41
  propriétés algébriques, 47

représentation, 48
représentation convexe, 49
simple, 3

local feature size, 71

médian généralisé, *voir* center point
maillage, *voir* complexe cellulaire
matrice
adjacence, 47
incidence, 47
laplacienne, 47, 51

optimisation
problème d', 96

polyèdre, 41
polytope, 17
position générale, 28
prédicat, 24
problème
d'optimisation, 96
hamiltonian cycle, 97
localisation planaire, 11
range search, 88
voyageur de commerce, *voir* TSP
PTAS, 97

range search, 88
reach, 71
reconstruction, 69
multi-échelles, 75
représentation
énérgie, 53
barycentrique, 51

séparateurs, 59
algorithme de Lipton-Tarjan, 60
codage de graphes, 65
définition, 59
graphes géométriques, 62
graphes planaires, 60
localisation planaire, 64
Schnyder
algorithme de dessin, 55
défintion d'une 3-orientation, 55
forêts, 55
simplexe
définition, 5
de Delaunay, 28
squelette, 70
structure de données
kd-trees, 92
à base de triangles, 10
Half-edge, 10
Quad-edge, 10
range trees, 89
triangulations hiérarchiques, 86

témoins, 76
complexe de, 76
théorème
Carathéodory, 15
Center Point, 17
de la borne supérieure, 17
des témoins, 76
Fáry, 41
formule d'Euler, 7
Helly, 15, 107
Koebe, 44
Kuratowski, 43
Radon, 14
reconstruction par complexe de témoins, 79
reconstruction par Delaunay restreint, 73
Steinitz, 41
Tutte, 49
Tverberg, 107
Whitney, 41
triangulation, 5
d'un ensemble de points, 5
définition, 27
de Delaunay, 28
de Delaunay restreinte, 73
planaire, 41
TSP, 97
Euclidean-TSP, 100
Metric-TSP, 98

Voronoï
diagramme, 29
diagramme restreint, 73