

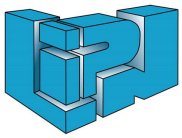


Université d'ANTANANARIVO
Domaine des Siences et Technologies
Mention Mathématiques et Informatique
MISA

En vue de l'obtention du diplôme de Master 2 en
Mathématiques Informatique et Statistiques Appliquées

Génération aléatoire de polytopes entiers

Présenté le 23 septembre 2016 par :
Rado Andrianandrasana RAKOTONARIVO

Devant le jury composé de :

Président du jury :	Arthur RANDRIANARIVONY
Examinateur :	Patrick RABARISON
Encadreur :	Vonjy RASENDRAHASINA
Co-encadreurs :	Lionel POURNIN
	Julien DAVID

Remerciements

Je tiens à remercier Julien David et Lionel Pournin pour leur aide et leur accueil chaleureux dans leur travail d'encadrement. Je leur exprime toute ma gratitude pour tout ce qu'ils m'ont appris et m'ont fait découvrir durant le stage et surtout pour le soutien qu'ils ont montré à mon égard lors de ces cinq mois de travail et bien avant ma venue en France.

Je tiens également à remercier Vlady Ravelomanana ainsi que Andry Rasoanaivo pour leur bienveillance et sans qui je n'aurai pu obtenir le stage.

Je voudrais aussi exprimer ma gratitude aux membres du jury de ma soutenance, Messieurs Arthur Randrianarivony, Patrick Rabarison et Vonjy Rasendrahasina, pour leur disponibilité ainsi que leur amabilité de bien avoir voulu examiner les résultats de mon travail.

J'exprime également toute ma gratitude ainsi qu'une sincère reconnaissance envers ma famille pour tout le soutien qu'ils ont apporté durant toutes mes années d'étude.

Puis, un grand merci à tous mes collègues à la MISA, en particulier à Tsinjo Tony Rakotoarimalala pour son aide et son soutien lors de mon séjour en France.

Et enfin, et non le moindre, je tiens à rendre gloire à Dieu pour tout ce qui a été fait et pour tout ce qui reste à venir. A Lui seul la gloire.

Table des matières

Introduction

Introduction générale

Des polytopes...

Les polytopes sont des enveloppes convexes d'ensemble finis de points dans l'espace Euclidien $\mathbb{E}^d$ [15]. Dans le plan, un triangle est un exemple de polytope. Un polytope est dit entier si ses sommets sont à coordonnées entières. De par leur étude très ancienne, puisqu'Euclide en a commencé le traitement mathématique (vers 300 av J.C.), l'intérêt pour les polytopes a connu un développement important grâce à l'essor de l'informatique. Un des exemples de l'utilisation de ces objets est donné par leur place importante dans la résolution de problème d'optimisation mathématique. Un polytope est en effet le domaine admissible compact d'un programme linéaire. Les polytopes entiers font également partie des objets combinatoires les plus utilisés en géométrie discrète.

Malgré cela, les polytopes, les propriétés de leurs graphes et, en général, leur combinatoire restent encore mal connus à ce jour [15] – le graphe d'un polytope étant son 1-squelette. Par exemple, le théorème de Steinitz selon lequel un graphe est le graphe d'un polytope de dimension 3 si et seulement si il est planaire et 3-connexe, donne une caractérisation du graphe des polytopes de dimension 3 alors qu'il n'y a aucune caractérisation connue pour les polytopes de dimension supérieure. Ce qui ne fait que renforcer la motivation

pour leur étude.

...à la génération aléatoire

Une classe combinatoire $\mathcal{C}$ est un ensemble fini d'objets, munie d'une fonction taille tel qu'il existe un nombre fini d'objets de chaque taille. Ces objets sont alors des objets combinatoires. On associe, en particulier, à toute classe combinatoire $\mathcal{C}$ une série génératrice $C(z) = \sum_{n\geq 0} c_n z^n$, dont le coefficient $c_n = [z^n]C(z)$ énumère les structures de taille n [9]. Ces séries tiennent une place essentielle dans les traitements liés à ces structures combinatoires.

Un générateur d'objets combinatoires est un algorithme permettant d'engendrer ces objets selon une distribution de probabilité préalablement définie. On s'intéresse le plus souvent à la distribution uniforme sur des objets de même taille. C'est-à-dire que la probabilité de tirer un objet de taille n est de $\frac{1}{c_n}$ avec c_n fini. Engendrer aléatoirement des objets combinatoires permet, non seulement, de mener des études expérimentales, puis d'émettre ou d'infirmer des conjectures sur les propriétés des objets engendrés, mais également de tester l'efficacité des algorithmes qui prennent ces objets en paramètres.

Objectifs du stage

Le stage que j'ai effectué a pour but d'engendrer aléatoirement des polytopes entiers afin d'en étudier les propriétés. L'un des objectifs était de développer des méthodes efficaces pour la génération aléatoire de polytopes entiers. La distribution étudiée étant l'uniforme parmi les polytopes de dimension d contenus dans l'hypercube de côté k et à v sommets. Cependant ma contribution s'est restreinte à la dimension $d = 2$, i.e. la génération de polygones v sommets dans $[0, k]^2$.

La méthodologie que l'on s'est proposé d'adopter a été l'étude expérimentale de ces questions avant de les aborder d'un point de vue théorique. L'étape expérimentale permet à la fois de prévoir les résultats à prouver, mais aussi d'obtenir de meilleures intuitions sur les méthodes de preuves efficaces.

Le contexte étant placé, ce rapport suivra le plan suivant. Après une brève

présentation des deux organismes, la MISA et le LIPN, qui m'ont permis de réaliser mon stage de recherche, une première partie mettra en lumière de plus amples détails sur les polytopes ainsi qu'un état de l'art concernant le sujet et les autres termes clés en rapport au stage. Une deuxième partie détaillera mon apport dans le développement du générateur aléatoire. Et une troisième partie portera sur les expérimentations réalisées ainsi que les résultats que nous avons pu obtenir au cours de ces cinq mois de travail. Enfin, ce document sera conclu par mes appréciations personnelles par rapport au stage ainsi que les perspectives pour la continuité du travail réalisé jusqu'alors.

Les organismes du stage

La MISA

La **MISA** ou *Mathématiques, Informatique et Statistiques Appliquées* est une formation créée en 1996 au Département Mathématiques et Informatique (DMI) de la Faculté des Sciences de l'Université d'Antananarivo, avec l'aide du projet PRESUP (Programme de Renforcement de l'Enseignement Supérieur) de la Coopération Française. Elle délivre une Licence en Mathématique, Informatique et Statistiques appliquées ainsi qu'un Master à double compétence M2PR (Master 2 professionnel et recherche) depuis son adoption du système LMD en 2015.

D'une part, la MISA forme des cadres aptes à s'intégrer dans le monde de l'entreprise, et d'autre part, initie des jeunes chercheurs dans les domaines de l'informatique et des statistiques.

A la fin de son cursus, chaque étudiant en Master 2 est amené à effectuer un stage d'une durée de six mois en entreprise ou dans un laboratoire de recherche selon l'option choisie. Ayant choisi l'option recherche j'ai effectué mon stage au sein du Laboratoire d'Informatique de Paris-Nord.

Je tiens à préciser qu'avant de travailler sur ce stage j'ai déjà commencé à travailler sur des jeux de coloration d'hypergraphes ainsi que des constructions de modèles de vote avec Vlady Ravelomanana ce qui explique pourquoi je ne me suis concentré sur ce stage que pendant cinq mois.

Le LIPN

Le **LIPN** ou *Laboratoire d'Informatique de Paris-Nord* est un laboratoire de recherche en informatique au sein de l'Institut Galilée de l'Université Paris 13. Il est associé au Centre National de la Recherche Scientifique (CNRS) depuis 1992 et a un statut d'unité mixte de recherche (UMR 7030) depuis 2001. Le LIPN poursuit ses recherches autour de plusieurs axes forts en s'appuyant sur les compétences de ses membres, en particulier en combinatoire, en optimisation combinatoire, en algorithmique, en logique, en génie logiciel, en langage naturel, en apprentissage. Le laboratoire est structuré en cinq équipes suivant toujours ces axes :

— L'équipe A3 : Apprentissage Artificiel et Applications
— L'équipe AOC : Algorithmes et Optimisation Combinatoire
— L'équipe CALIN : Combinatoire, ALgorithmique et INteractions
— L'équipe LCR : Logique, Calcul et Raisonnement
— L'équipe RCLN : Représentation des Connaissances et Langage Naturel

Plus de 150 membres participent aux activités du laboratoire dont 87 chercheurs ou enseignants-chercheurs permanents, pour la plupart en poste à l'Université Paris 13 soit à l'Institut Galilée, soit à IUT de Villetaneuse. Le LIPN fait partie du pôle MathSTIC, un centre de recherche dans les domaines mathématiques et sciences et technologies de l'information et de la communication.

Durant mon stage, j'ai été affecté dans l'équipe CALIN. L'équipe CALIN réunit des chercheurs ayant des compétences dans les divers domaines de la combinatoire (analytique, algébrique et bijective) avec un intérêt pour la complexité des algorithmes dont ils cherchent à déterminer le comportement en moyenne ou en distribution, pour l'analyse fine de structures de données, ainsi que pour la physique. L'équipe s'organise autour de deux sous-axes : l'un tourné vers la physique combinatoire, l'autre abordant l'analyse d'algorithmes et de structures combinatoires.

Première partie

Définitions et état de l'art

Chapitre 1

Polytopes

Les définitions et propriétés de ce chapitre sont issues de [12] et de [4].

1.1 Notion de convexité

Soit $\mathcal{A}$ un ensemble fini de points dans l'espace euclidien $\mathbb{E}^d$.

1.1.1 Combinaison linéaire

Définition 1.1. *Une combinaison linéaire d'éléments de $\mathcal{A}$ est une somme $\sum_{i=1}^{k} \lambda_i a_i$ où k appartient à $\mathbb{N}$, et pour tout i allant de 1 à k, les λ_i sont des réels et les a_i des éléments de $\mathcal{A}$.*

1.1.2 Combinaison affine

Définition 1.2. *Une combinaison $\sum_{i=1}^{k} \lambda_i a_i$ est dite affine si on a $\sum_{i=1}^{k} \lambda_i = 1$.*

Définition 1.3. *L'ensemble des combinaisons affines d'éléments de $\mathcal{A}$ engendre un sous-espace affine de $\mathbb{R}^d$, un sous-espace qu'on appelle enveloppe affine de $\mathcal{A}$.*

1.1.3 Espace affine

Définition 1.4. *Un espace affine ou plus généralement un sous-espace affine de $\mathcal{A}$ est le sous-espace engendré par une famille finie de combinaisons affines de points de $\mathcal{A}$. En particulier ce sous-espace est appelé enveloppe affine de $\mathcal{A}$.*

Par exemple, l'enveloppe affine de deux points distincts est la droite qui passe par ces deux points. Plus généralement, $k+1$ points sont dits affinement indépendants s'ils engendrent un sous-espace affine de dimension k.

1.1.4 Combinaison convexe

Définition 1.5. *Une combinaison $\sum_{i=1}^{k} \lambda_i a_i$ est dite convexe si elle est affine et pour tout i de 1 à k, $\lambda_i \geq 0$.*

Définition 1.6. *Un ensemble $\mathcal{A}$ est dit convexe si pour tout couple de points de $\mathcal{A}$, le segment qui les joint est inclus dans $\mathcal{A}$.*

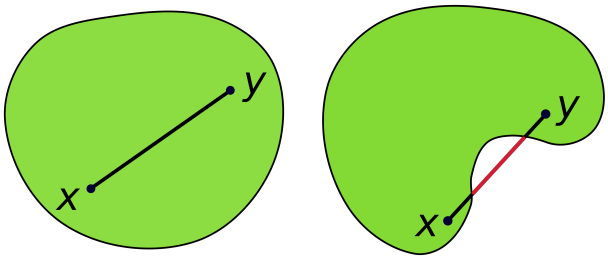

FIGURE 1.1 – A gauche une partie convexe, à droite une partie non convexe

Exemple 1.1. *Étant donnés trois points dans le plan, le point P est une combinaison convexe des trois points x_1, x_2, x_3, tandis que Q ne l'est pas. Q est seulement une combinaison affine des trois points. Voir la figure 1.2.*

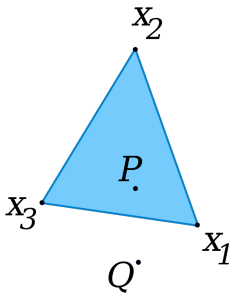

FIGURE 1.2 – Une combinaison convexe

1.1.5 Enveloppe convexe

Définition 1.7. *L'enveloppe convexe d'un ensemble $\mathcal{A}$, noté Conv($\mathcal{A}$) est le plus petit convexe contenant $\mathcal{A}$. De plus, Conv($\mathcal{A}$) est l'ensemble des combinaisons convexes de familles finies de points de $\mathcal{A}$.*

1.2 Polytopes

1.2.1 Polytopes

Définition 1.8. *Un polytope est l'enveloppe convexe d'un ensemble fini de points de $\mathbb{E}^d$. Sa dimension d est celle du sous-espace affine qu'il engendre. On note un polytope de dimension d un* d-polytope.

Définition 1.9. *On appelle simplexe ou d-simplexe l'enveloppe convexe de $d+1$ points affinement indépendants. Les d-simplexes sont donc exactement les polytopes de dimension d avec $d+1$ sommets.*

Exemple 1.2. *L'enveloppe convexe de deux points P et Q dans le plan est le segment $[PQ]$, celle de trois points P, Q et R est le triangle PQR. Le segment $[PQ]$ et le triangle PQR sont respectivement des 1-simplexe et 2-simplexe.*

1.2.2 Faces d'un polytope

Soit P un polytope de dimension d.

Définition 1.10. *Soit $\mathbb{E}$ l'espace ambiant de dimension d, on appelle hyperplans de $\mathbb{E}$ les sous-espaces de dimension $d-1$. Un hyperplan H divise $\mathbb{E}^d$ en deux sous-espaces.*

$$\mathbb{E}^d = H^+ \cup H \cup H^-.$$

Définition 1.11. *Une face de P est soit P lui-même soit l'intersection de P avec un hyperplan valide.*

Un tel hyperplan, aussi appelé support de la face, est un hyperplan H tel que P $\cap$ H soit non vide et que P soit entièrement inclus dans l'un des deux demi-espaces fermés $\overline{H^+}$ ou $\overline{H^-}$.

Définition 1.12. *La dimension d'une face est celle de son enveloppe affine.*

- *Une face de dimension 0 est appelée un sommet.*
- *Une face de dimension 1 est appelée une arête.*
- *Une face de dimension k est appelée une k-face.*
- *Une face de dimension dim(P)−1 est appelée une facette.*

On remarque que chaque face de P est un polytope.

Exemple 1.3. *Soit P un cube.*

- *P est un polytope de dimension 3*
- *P possède 6 facettes (face de dimension 2)*
- *P possède 12 arêtes (face de dimension 1)*
- *P possède 8 sommets (face de dimension 0)*

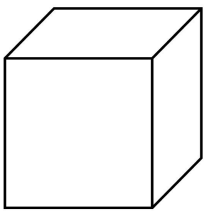

FIGURE 1.3 – Un cube

Définition 1.13. *Le treillis des faces d'un polytope P est le treillis dont les éléments sont les faces et dont la relation de couverture/ordre est l'inclusion.*

1.2.3 Graphe d'un polytope

Définition 1.14. *Le graphe d'un polytope est le graphe dont les sommets sont ceux du polytope et les arêtes sont les faces de dimension 1 du polytope.*

Le graphe d'un polytope fait référence à son 1-squelette. Le k-squelette d'un polytope est fait des faces de dimension au plus k, donc le 1-squelette est fait des sommets et des arêtes.

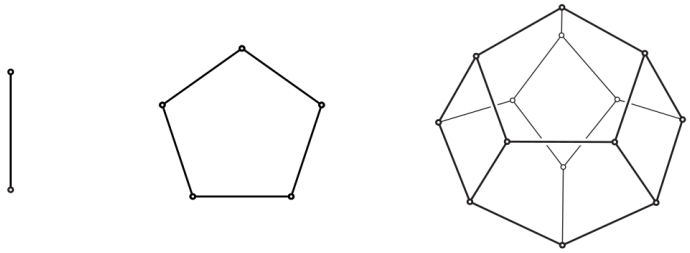

FIGURE 1.4 – Graphes de polytopes : un segment (polytope de dimension 1), un pentagone (polytope de dimension 2), un associaèdre de dimension 3 (polytope de dimension 3)

1.2.4 Polytopes entiers

Définition 1.15. *Un polytope P est dit entier si tous ses sommets sont à coordonnées entières.*

Le stage s'est exclusivement porté sur l'étude des polytopes entiers. Dans toute la suite de ce document, on comprendra par polytope un polytope entier. Le nombre de polytopes entiers de dimension d à v sommets inclus dans $[0, k]^d$ est fini. De ce fait on peut considérer ces polytopes comme étant des objets combinatoires.

1.3 Calcul d'enveloppe convexe en dimension 2

Puisque mon travail durant le stage s'est limité aux polytopes de dimension $d = 2$, je vais présenter deux algorithmes de calcul d'enveloppe convexe d'un ensemble de points dans le plan.

Soit $\{x_0, x_1, ..., x_n\}$ un ensemble de n points dans le plan. Calculer l'enveloppe convexe de cet ensemble de points revient alors à trouver les h points (où $h \leq n$) tel que ces h points soient les sommets du polygone convexe qui enveloppe tous les points de l'ensemble.

1.3.1 Algorithme de Jarvis

L'algorithme de Jarvis également appelé "technique du papier cadeau" permet de calculer avec un temps relativement efficace une enveloppe convexe.

Principe

On commence le calcul en partant d'un point extrême x_0, le plus souvent celui d'absisse minimale. Ce point fait partie de l'enveloppe convexe. Puis on recherche le point x_1 tel que l'angle x_0x_1 avec l'horizontale soit minimum. Puis on itère la recherche du point x_i tel que l'angle $x_{i-1}x_i$ avec $x_{i-2}x_{i-1}$ soit minimum jusqu'à ce que l'on revienne au point x_0.

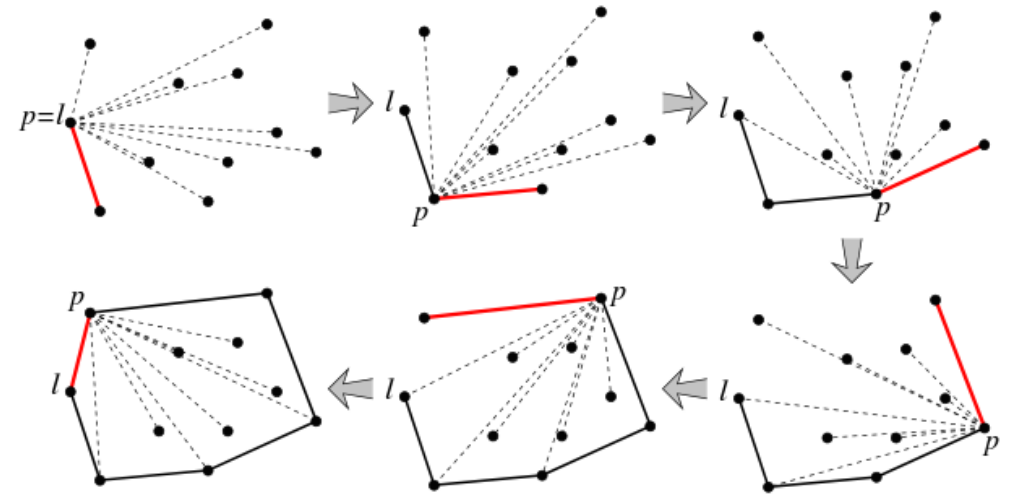

FIGURE 1.5 – L'algorithme de Jarvis - Figure tirée de [8]

Complexité

L'algorithme de Jarvis est un algorithme que l'on qualifie de « sensible à la sortie ». En effet, sa complexité d'exécution en temps est de $\mathcal{O}(nh)$ où h est le nombre de sommets de l'enveloppe convexe. Et au pire cette complexité est de l'ordre de $\mathcal{O}(n^2)$.

Cet algorithme est donc très efficace lorsque l'enveloppe convexe ne comprend que peu de sommets.

1.3.2 Parcours de Graham

Le parcours de Graham quant à lui permet de réduire la complexité du calcul à l'ordre de $\mathcal{O}(n \log n)$ ce qui fait son intérêt principal. Son principe est le suivant :

Principe

1. Chercher le point x_0 de plus petite ordonnée. S'il y a égalité entre un ou plusieurs points, choisir x_0 comme étant celui de plus petite abscisse.
2. Trier ensuite l'ensemble des n points dans un tableau T en fonction de l'angle que chacun d'entre eux fait avec l'axe des abscisses relativement à x_0 (il n'est pas nécessaire de calculer l'angle réel ; on peut se limiter à la comparaison des tangentes, ou bien même utiliser le produit en croix des coordonnées pour connaître les positions relatives des points). Voir la figure 1.6.
3. Considérer 3 éléments (p : *précédent*, c : *courant*, s : *suivant*) contigus de T, vus comme deux couples successifs. Pour chacune de ces paires de couples, si passer du premier couple au second constitue un « tournant à droite », rejeter c de T. Sinon si c'est un « tournant à gauche », le garder et passer au point suivant de T. Voir la figure 1.7.
4. Répéter le procédé jusqu'à ce que l'algorithme revienne à x_0.
5. L'enveloppe convexe est alors constitué des points de T.

En image, cela donne :

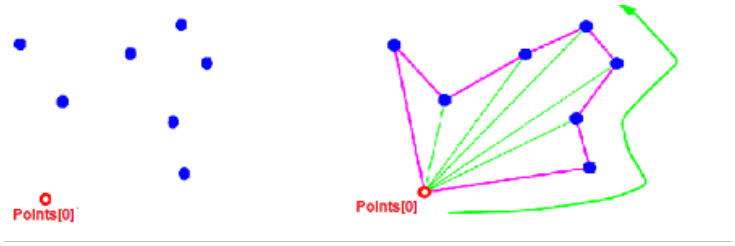

FIGURE 1.6 – Parcours de Graham - Tri des points suivant l'angle avec l'axe des abscisses relativement à x_0 - Figure tirée de [13]

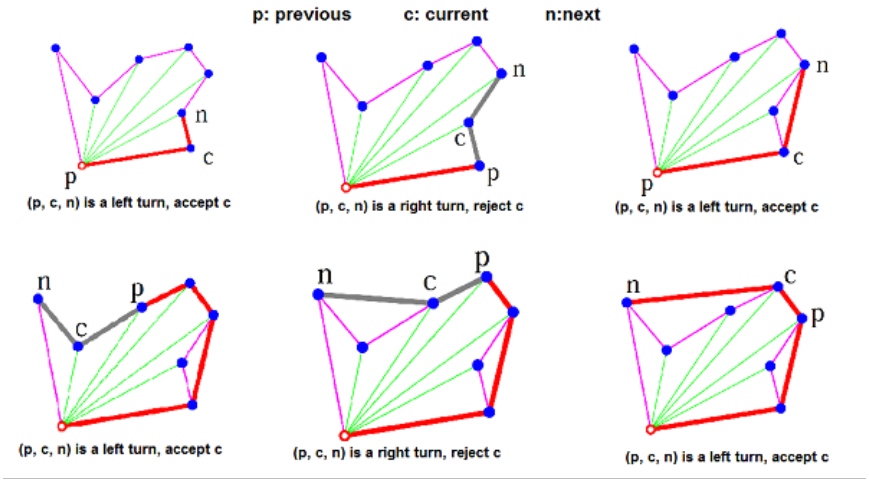

FIGURE 1.7 – Parcours de Graham - Rejet des points formant une concavité - Figure tirée de [13]

Complexité

Pour les n points en entrée, la complexité de l'algorithme est de $\mathcal{O}(n \log n)$ si on utilise un tri en $\mathcal{O}(n \log n)$.

En effet on trouve x_0 en $\mathcal{O}(n)$. Ensuite le tri se fait en $\mathcal{O}(n \log n)$. Le test de rejet ou d'acceptation se fait également en $\mathcal{O}(n)$ car pour chaque couple, on ne teste que le rejet ou l'acceptation de c. Donc la complexité de l'algorithme est de $\mathcal{O}(n \log n)$.

Une implémentation de cet algorithme sera donnée dans la séction 4.2.

Chapitre 2

Génération aléatoire

2.1 Génération aléatoire

2.1.1 Classe combinatoire

Définition 2.1. *Une classe combinatoire $\mathcal{C}$ est un ensemble fini ou dénombrable sur lequel on définit une fonction qui vérifie les deux propriétés suivantes :*

— *La taille d'un élément est un entier positif ou nul.*

$$\text{taille} : \mathcal{C} \to \mathbb{N}.$$

— *Il n'y a qu'un nombre fini d'objet de chaque taille.*

$$\forall i \in \mathbb{N}, \{o | \text{taille}(o) = i\} \textit{est fini.}$$

Deux classes combinatoires sont *isomorphes* si elles contiennent exactement le même nombre d'objets de chaque taille. Cela revient à dire qu'il existe une bijection entre les deux classes qui préserve la taille.

2.1.2 Série génératrice ordinaire

Définition 2.2. *La série génératrice associée à une classe $\mathcal{C}$ est définie par :*

$$C(z) = \sum_{\alpha \in \mathcal{C}} z^{|\alpha|} = \sum_{n \geq 0} c_n z^n \textit{ avec } c_n = Card\{\alpha \in \mathcal{C}, |\alpha| = n\}.$$

On considère ces séries comme des séries formelles dont on veut extraire des informations. $[z^n]C(z)$ désigne le coefficient c_n de z^n dans $C(z)$.

Exemple 2.1.

— *La série génératrice ordinaire des mots binaires est*

$$B(z) = \sum_{n\geq 0} 2^n z^n = \frac{1}{1-2z}.$$

Il existe exactement 2^n mots binaires de taille n. Ici, la taille porte sur la longueur du mot.

— *La série génératrice ordinaire des entiers positifs est*

$$I(z) = \sum_{n\geq 1} 1z^n = \frac{1}{1-z}$$

Il existe exactement 1 entier positif de taille n.

2.1.3 Génération aléatoire

Soit donc $\mathcal{C}$ une classe combinatoire. Faire de la génération aléatoire c'est vouloir tirer «au hasard» un élément dans cette classe. Le terme «au hasard» sous entend une distribution de probabilité que l'on définit au préalable. On considère dans la plupart des cas l'uniforme, ce qui est aussi le cas dans tout ce qui va suivre dans ce document. On souhaite alors que tous les éléments de l'ensemble aient la même probabilité d'être tirés.

Comme chaque classe combinatoire a une série génératrice qui lui est associée, la probabilité de tirer un élément γ de $\mathcal{C}$ de taille n serait :

$$P(\gamma) = \frac{1}{c_n},$$

où c_n désigne le nombre d'élement de taille n de $\mathcal{C}$.

Exemple 2.2. *Pour la classe combinatoire des mots binaires. On a $B(z) = \sum_{n\geq 0} 2^n z^n$. La probabilité de tirer un mot binaire de taille 3 est de $\frac{1}{2^3} = \frac{1}{8}$. En effet cela revient à tirer un mot parmi $\{000, 001, 010, 100, 011, 101, 110, 111\}$.*

Que ce soit pour tester la robustesse d'un programme, vérifier l'adéquation d'un modèle ou confirmer une conjecture, engendrer aléatoirement des objets combinatoires permet d'effectuer des simulations afin d'obtenir des données statistiques sur des structures le plus souvent abstraites. L'objet de ces simulations est principalement d'engendrer des structures suffisamment grandes pour rendre observables leurs propriétés limites, d'où la nécessité de concevoir des générateurs efficaces.

Générateurs ad-hocs

Les générateurs ad-hocs sont des générateurs dédiés pour des classes combinatoires spécifiques. Ces générateurs exploitent les propriétés structurelles d'une classe donnée pour obtenir des méthodes de génération performantes.

Méthodes génériques

Les générateurs automatiques quant à eux sont des générateurs plus globaux qui ont souvent pour base commune la méthode symbolique. La méthode symbolique permet en effet de passer des constructions assemblistes aux séries génératrices de manière mécanique pour la classe combinatoire étudiée. La méthode symbolique ainsi que les constructions combinatoires qui vont suivre sont détaillées dans [9].

Prenons par exemple une classe $\mathcal{C}$ qui serait une union disjointe de classes $\mathcal{A}$ et $\mathcal{B}$ dont les séries génératrices $A(z)$ et $B(z)$ sont connues. Avec la génération automatique, on peut obtenir la série génératrice de $\mathcal{C}$ à partir de ceux de $\mathcal{A}$ et de $\mathcal{B}$. Une telle classe $\mathcal{C}$ est dite une constructible ou spécifiable car elle peut être décrite par le moyen de constructions combinatoires et/ou de classes primitives.

Définition 2.3. *Une construction combinatoire est admissible si le nombre d'éléments résultant de la construction ne dépend que du nombre d'éléments des classes à partir desquelles la construction a été faite.*

1. Les classes primitives

   Classe neutre : C'est la classe $\mathcal{E}$ qui admet un seul élément de taille 0. Sa série génératrice est :

   $$E(z) = z^0 = 1.$$

   Classe atomique : C'est la classe $\mathcal{Z}$ constituée d'un élément de taille 1. Sa série génératrice est :

   $$Z(z) = z^1 = z.$$

2. Les constructions combinatoires usuelles.

   Soient $\mathcal{A}$ et $\mathcal{B}$ et leurs séries génératrices $A(z)$ et $B(z)$ et $\mathcal{C}$ la classe à spécifier.

   Union disjointe $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ et $\mathcal{A} \cap \mathcal{B} = \emptyset$. La taille sur $\mathcal{C}$ d'un élément γ est :

   $$|\gamma|_{\mathcal{C}} = \begin{cases} |\gamma|_{\mathcal{A}} & si \gamma \in \mathcal{A} \\ |\gamma|_{\mathcal{B}} & si \gamma \in \mathcal{B} \end{cases}$$

   Donc $c_n = a_n + b_n$ i.e.

   $$\sum_{n \geq 0} c_n z^n = \sum_{n \geq 0} a_n z^n + \sum_{n \geq 0} b_n z^n.$$

   Produit cartésien $\mathcal{C} = \mathcal{A} \times \mathcal{B} = \{\gamma = (\alpha, \beta) | \alpha \in \mathcal{A} \; et \; \beta \in \mathcal{B}\}$ où la taille sur $\mathcal{C}$ de γ est :

   $$|\gamma|_{\mathcal{C}} = |\gamma|_{\mathcal{A}} \times |\gamma|_{\mathcal{B}}$$

   Et $c_n = \sum_{j=0}^{n} a_j b_{n-j}$. On a :

   $$\sum_{n \geq 0} c_n z^n = \sum_{n \geq 0} \sum_{j=0}^{n} a_j b_{n-j} z^n = \sum_{k \geq 0} a_k z^k \times \sum_{k \geq 0} b_k z^k$$

   Séquences $\mathcal{C} = Seq(\mathcal{A}) = \{\mathcal{E}\} \bigcup \mathcal{A} \bigcup \mathcal{A} \times \mathcal{A} \bigcup \mathcal{A} \times \mathcal{A} \times \mathcal{A} ...$

   $$\mathcal{C} = \{\gamma = (\alpha_1, \alpha_2, ..., \alpha_l) | l \geq 0 \; et \; 1 \leq i \leq l, \alpha_i \in \mathcal{A}\},$$

$\mathcal{C}$ ne doit contenir qu'un nombre fini d'objets de chaque taille. Pour cela $\mathcal{A}$ ne doit pas contenir d'objet de taille 0, ie $a_0 = 0$. La taille sur $\mathcal{C}$ de γ est :

$$|\gamma|_{\mathcal{C}} = \sum_{i=1}^{l} |\alpha_i|_{\mathcal{A}},$$

$$C(z) = 1 + A(z) + A(z)^2 + A(z)^3 + ... = \sum_{i \geq 0} A(z)^i,$$

$$C(z) = \frac{1}{1 - A(z)}.$$

Méthodes récursives

La génération récursive fait partie des méthodes de génération génériques. Elle consiste à trouver le nombre d'éléments de taille n en fonction du nombre d'éléments de taille inférieure. Cette méthode nécessite donc un pré-calcul du nombre d'éléments de taille inférieure pour obtenir le nombre d'éléments de taille n.

Exemple 2.3. *Une définition récursive d'une série génératrice est de la forme :*

$$T(z) = \sum_{n \geq 0} t_n z^n \text{ où } t_n = \alpha t_{n-\beta} + \gamma,$$

les coefficients α, β et γ sont des entiers positifs ou nuls.

Le point essentiel pour la génération récursive réside donc dans le pré-calcul à effectuer. La génération de polygones convexes effectuée durant le stage s'est centrée sur cette méthode de génération. Nous avons essayé de faire ce pré-calcul de deux manières différentes :

1. La construction d'un générateur exhaustif qui a pour objectif d'énumérer tous les éléments de chaque taille, qui représente un calcul assez lourd.
2. Une estimation du nombre d'éléments de chaque taille.

Ces deux approches seront détaillées plus tard dans le document dans les chapitres 5 et 6.

2.2 Génération aléatoire de polygones convexes

Du fait que les polytopes entiers contenus dans l'hypercube $[0, k]^d$ soient des objets combinatoires, des travaux sur la génération aléatoire de polygones ont été faits. Il sera question dans cette section de travaux s'y rapportant.

2.2.1 Génération aléatoire de polygones convexes dans un disque

Cette section présente une méthode de génération aléatoire de polygones convexes dans un disque. L'algorithme engendre un polygone aléatoire défini par n points aléatoires uniformément distribués dans un disque sans avoir à générer explicitement tous les points. Pour de plus amples précisions sur l'algorithme, voir [7].

L'idée conductrice de l'algorithme se base sur la génération d'un petit polygone P_i avec i points dans un disque $\mathcal{D}$, puis augmenter progressivement ensuite le nombre de points jusqu'à obtenir P_n. Son principe est le suivant :

1. Générer un petit nombre initial de points dans $\mathcal{D}$ et calculer son enveloppe convexe.
2. Calculer le rayon du plus grand disque $\mathcal{D}_i$ inscrit dans l'enveloppe convexe.
3. Simuler un nombre de m_i points qui vont être tirés à l'intérieur du disque inscrit $\mathcal{D}_i$.
4. Tirer les $n - m_i$ points restant dans l'anneau défini par $\mathcal{D} - \mathcal{D}_i$ et mettre à jour l'enveloppe convexe. Voir la figure 2.1.
5. Répéter jusqu'à ce que la somme du nombre de points générés et du nombre de points simulés soit égale à n.

Le point fort de cette méthode se base sur la simulation et non la génération explicite de points dans le disque. Avec cette approche il est possible de limiter, d'une certaine façon, voire d'éviter de tirer des points à l'intérieur de l'enveloppe convexe.

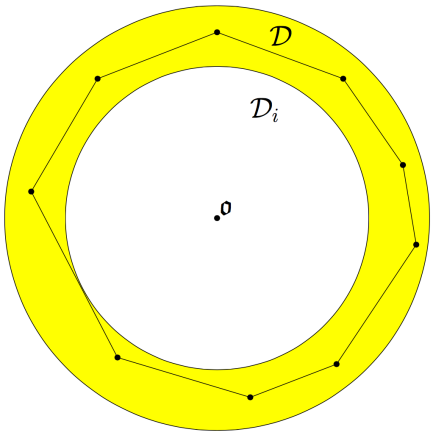

FIGURE 2.1 – À l'étape i, simuler un nombre m_i de points tirés à l'intérieur de $\mathcal{D}_i$ et générer uniformément les $n - m_i$ points dans l'anneau jaune. Figure tirée de [7].

2.2.2 Génération aléatoire de polyominoes digitalement convexes

Il s'agit d'un article qui présente les résultats du travail de Olivier Bodini, Alice Jacquot, Philipe Duchon et de Ljuben R. Mustafchiev [3] sur l'analyse asymptotique et génération aléatoire de polyominoes digitalement convexes. Cet article a été intéressant dans la mesure où ils ont trouvé une caractérisation de polyominoes digitalement convexes en utilisant la combinatoire des mots par des *mots de Christoffel* ainsi qu'une génération uniforme assez efficace avec un générateur de Boltzmann.

Un *polyomino digitalement convexe*, abrégé en DCP est l'ensemble de toutes les cellules de $\mathbb{Z}^2$ contenues dans une partie convexe bornée du plan. On s'intéressera plutôt au contour du polyomino. Chaque DCP peut être décomposé en une chaîne de quatre sous-chemins WN, NE, ES et SW. Les quatre points W, N, E, S sont placés de la manière suivante :

— W est le point le plus bas sur la bordure gauche.
— N est le point le plus à gauche sur la bordure haute.
— E est le point le plus haut sur la bordure droite.
— S est le point le plus à droite sur la bordure basse.

Chaque sous-chemin WN, NE, ES et SW n'autorise que des pas de direction *nord* et *est* et sont dits NW-convexe. Donc tirer aléatoirement un DCP consiste à engendrer quatre chemins NW-convexes et les relier ensuite.

Ils ont ensuite encodé les chemins NW-convexes par des mots de Christoffel dont la génération aléatoire est connue.

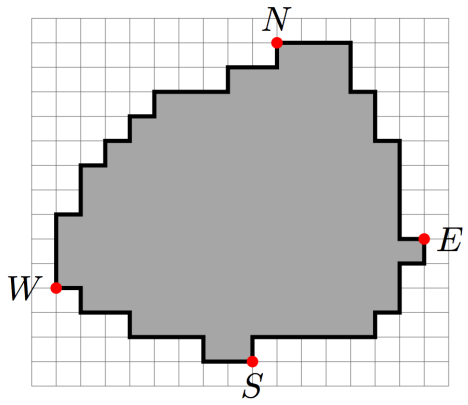

FIGURE 2.2 – Un polyomino composé de quatre chemins NW-convexes. Le chemin WN est encodé par le mot w =1110110101010001001. Figure tirée de [3]

De cette manière, engendrer aléatoirement un DCP consiste à tirer les quatre sous-chemins WN, NE, ES et SW en utilisant des appels indépendants d'un générateur de Boltzmann à paramètre x. Puis, si la longueur $|WN|+|NE| = |ES|+|SW|$ et si $|NE|+|ES| = |SW|+|WN|$ on retourne le DCP (WN, NE, ES, SW).

Cette méthode de génération montre une approche intéressante dans la génération de polygones convexes. Nous avons construit un générateur suivant à peu près le même principe.

Chapitre 3

Polymake

3.1 Présentation de polymake

Cette section aura pour but de présenter la philosophie globale de l'outil `polymake` mais aussi d'avoir un petit aperçu de ce dont il est capable de faire. Bien que `polymake` soit capable de traiter un grand nombre d'objets géométriques, on se limitera plutôt au traitement des polytopes. Et enfin, voir en quoi `polymake` est intéressant par rapport au travail que j'ai effectué.

`polymake` est un logiciel mathématique très complet qui fournit une large palette de fonctions pour les polytopes convexes, les complexes simpliciaux, et d'autres objets géometriques [11]. Outre les calculs appliqués à ces objets, `polymake` permet d'en avoir également une visualisation. Il permet par exemple de donner le nombre de face de chaque dimension pour un polytope donné. C'est un logiciel open source et facile d'utilisation disponible en téléchargement sur `https://polymake.org/doku.php/download/start`.

La philosophie principale de `polymake` est de servir d'interface entre l'utilisateur et les nombreux outils de traitement sur les polytopes qu'il regroupe. Une fois que l'utilisateur lui a décrit un polytope, il peut envoyer des requêtes au logiciel pour tirer des propriétés sur le polytope ainsi décrit. Par ailleurs `polymake` permet aux utilisateurs avancés d'étendre ses fonctionnalités pour des requêtes spécifiques de différentes manières.

Le logiciel offre beaucoup de possibilités quant à l'étude des polytopes

aux dimensions $d > 2$. Et du fait que le stage s'est restreint à la génération de polygones convexes, la familiarisation avec `polymake` a permis d'avoir une idée de ce qui se passe en dimension supérieure.

Les exemples de cette section ont été tirés du site de `polymake` ainsi que de [11].

3.2 Un exemple simple : le 3-cube

Supposons que l'on ait un ensemble S constitué d'un nombre fini de points dans un espace $\mathbb{R}^d$. Soit P leur enveloppe convexe, P est notre polytope. On veut à présent savoir quel est le nombre de facettes de P. Par exemple, soit $S = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$ dans $\mathbb{R}^3$. S est l'ensemble de sommets d'un cube.

Déclarer un tel polytope se traduit dans `polymake` par :

```
polytope > $p=new Polytope<Rational>;

polytope > $p->POINTS=<<".";
polytope (2)> 1 0 0 0
polytope (3)> 1 1 0 0
polytope (4)> 1 0 1 0
polytope (5)> 1 1 1 0
polytope (6)> 1 0 0 1
polytope (7)> 1 1 0 1
polytope (8)> 1 0 1 1
polytope (9)> 1 1 1 1
polytope (10)> .
```

On retrouve les coordonnées de nos sommets, une colonne de 1 supplémentaire est ajoutée par `polymake` pour homogénéiser son système de coordonnées. Faisons maintenant une requête à polymake pour savoir le nombre de facettes de P (ses faces de dimension 2) ; et si oui ou non il est simple.

Rappelons que P est simple si et seulement si chacun de ses sommets est contenu dans d facettes $d = dim(P)$.

```
polytope > print $p->N_FACETS;
6
polytope > print $p->SIMPLE;
1
```

On retrouve le nombre de facettes d'un cube en dimension 3, ainsi que la réponse 1 (`true`) à la requête `SIMPLE`. En effet chaque sommet de P est contenu dans exactement 3 facettes.

Selon la description faite du polytope, `polymake` décide de l'algorithme de calcul d'enveloppe qu'il utilisera. Cependant il est possible de spécifier la méthode à utiliser.

`polymake` est aussi capable de construire des polytopes standards. Par exemple pour le 3-cube, on aurait pu utiliser la commande suivante (le dernier argument 0 indique qu'il s'agit du cube unité.)

```
polytope > $p = cube(3,0);
```

3.3 Visualisation d'un polytope aléatoire

Comme il a été mentionné précédemment, `polymake` permet d'avoir un rendu visuel de polytope. Soit donc Q l'enveloppe convexe d'un ensemble de 20 points aléatoires uniformément distribués dans la sphère unité de $\mathbb{R}^3$. Il suffit d'entrer une commande pour décrire un tel polytope au logiciel et une autre pour en avoir un rendu visuel. Ici encore un calcul implicite de l'enveloppe convexe est effectué par `polymake` . `polymake` choisit parmi une des fonctions de calcul d'enveloppe convexe comme `cdd` [10], `lrs` [1], `porta` [6], et `qhull` [2].

```
polytope > $q = rand_sphere(3,20);

polytope > $q->VISUAL;
```

L'outil de visualisation standard de polymake est `JavaView`. Il est totallement interactif et permet de faire des rotations et/ou des agrandissements sur le rendu.

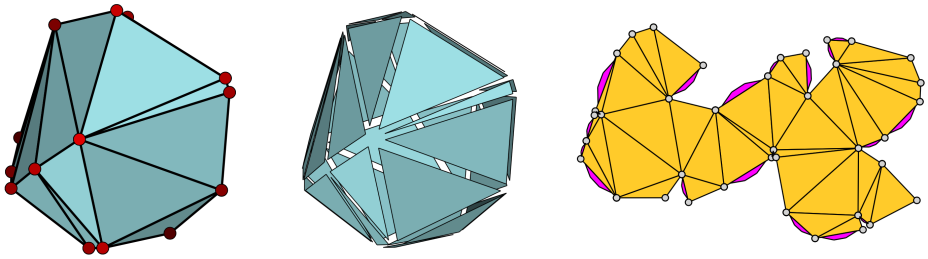

FIGURE 3.1 – Un 3-polytope à 20 facettes

3.4 Conclusion

Les exemples cités montrent toute l'utilité du calcul d'enveloppe convexe de manière efficace. Bien que durant le stage je me suis limité à la dimension $d = 2$, une prise en main rapide du logiciel m'a permis de visualiser les concepts abstraits liés au sujet en dimensions supérieures mais aussi de me poser des questions sur la génération aléatoire dans une sphère. Contacter les gens de `polymake` reste une perspective à l'aboutissement du travail dans la mesure où on pourrait leur proposer d'intégrer notre méthode de génération aléatoire de polytopes.

Deuxième partie

Contribution et travail effectué

Chapitre 4

Approche intuitive

Cette première section détaillera la méthode la plus intuitive pour générer aléatoirement des polygones convexes dans un carré de côté k. Le principe du générateur intuitif est simple. On tire au hasard n points dans le carré de côté k. On calcule ensuite l'enveloppe convexe des n points tirés avec le parcours de Graham et on a ainsi un polygone convexe. Bien évidement cette méthode naïve ne permet pas de paramétrer le nombre de sommets du polygone ainsi obtenu, de plus on ne peut pas contrôler la distribution de probabilité lors de la génération aléatoire du polygone. Néanmoins elle s'est avérée nécessaire pour la suite du travail dans la mesure où on a quand même pu tirer des propriétés intéressantes.

L'objectif reste toujours le même, engendrer aléatoirement des polygones convexe à v sommets dans $[0, k]^2$.

4.1 Algorithme naïf

Algorithm 1: Génération aléatoire d'un polygone convexe

Entrée: n : nombre de points, k : côté du carré

Sortie: Liste des sommets du polygone

```
1 ListeSommets ← NULL
2 ListePoints ← NULL
3 for i ← 0 to n − 1 do
4 |   genererAleatoirement(Point_i)
5 |   ListePoints.add(Point_i)
6 end
7 ListeSommets ← grahamScan(ListePoints)
8 return ListeSommets
```

4.2 Une implémetation du parcours de Graham

Algorithm 2: grahamScan

Entrée: Liste de points aléatoires

Sortie: Liste des sommets du polygone

```
1  ListeSommets ← NULL
2  Pivot
3  forall Point in ListePoints do
4  |  if Pivot_y < Point_y then
5  |  |  Pivot ← Point
6  |  end
7  end
8  trierParRapportAuPivot(Pivot, ListePoints)
9  forall triplet (previous, current, next) in ListePoints do
10 |  if tournantAGauche(previous, current, next) then
11 |  |  ListeSommets.add(current)
12 |  end
13 end
14 ListeSommets ← grahamScan(ListePoints)
15 return ListeSommets
```

Le point *Pivot* fait référence au point x_0 dans la description précédente de l'algorithme [Voir la section 1.3.2]. La fonction $tournantAGauche(previous, current, next)$ qui vérifie si le point current est à gauche du segment (previous,next) est décrite dans l'algorithme 3.

4.3 Résultats

Cette section présente un aperçu des résultats qu'on a pu obtenir en effectuant la génération de cette manière.

Algorithm 3: tournantAGauche($previous, current, next$)

Entrée: Point previous, Point current, Point next

```
1 P //previous
2 C //current
3 N //next
4 if determinant(PC, PN) > 0 then
5 |   return true
6 end
7 return false
```

4.3.1 Sorties

Un premier exemple d'exécution avec $n = 8$, $k = 10$ où le nombre de points générés n est petit. Voir la figure 4.1.

Un deuxième exemple avec $n = 100$, $k = 20$, où on a une valeur de n relativement grande, a donné le résultat suivant. Voir la figure 4.2.

4.3.2 Analyses

Après une série de générations aléatoires se basant sur cette méthode, on remarque que plus n augmente, plus le nombre de points tirés à l'intérieur de l'enveloppe convexe est grand ; c'est-à-dire que plus n est grand, plus le ratio de points retenus sur l'enveloppe convexe diminue [voir la figure 4.3 et la figure 4.4]. De plus nous n'avons aucun contrôle sur le nombre de sommets du polygone à la sortie, de ce fait on ne peut pas se baser sur cette méthode pour un travail sur les polytopes.

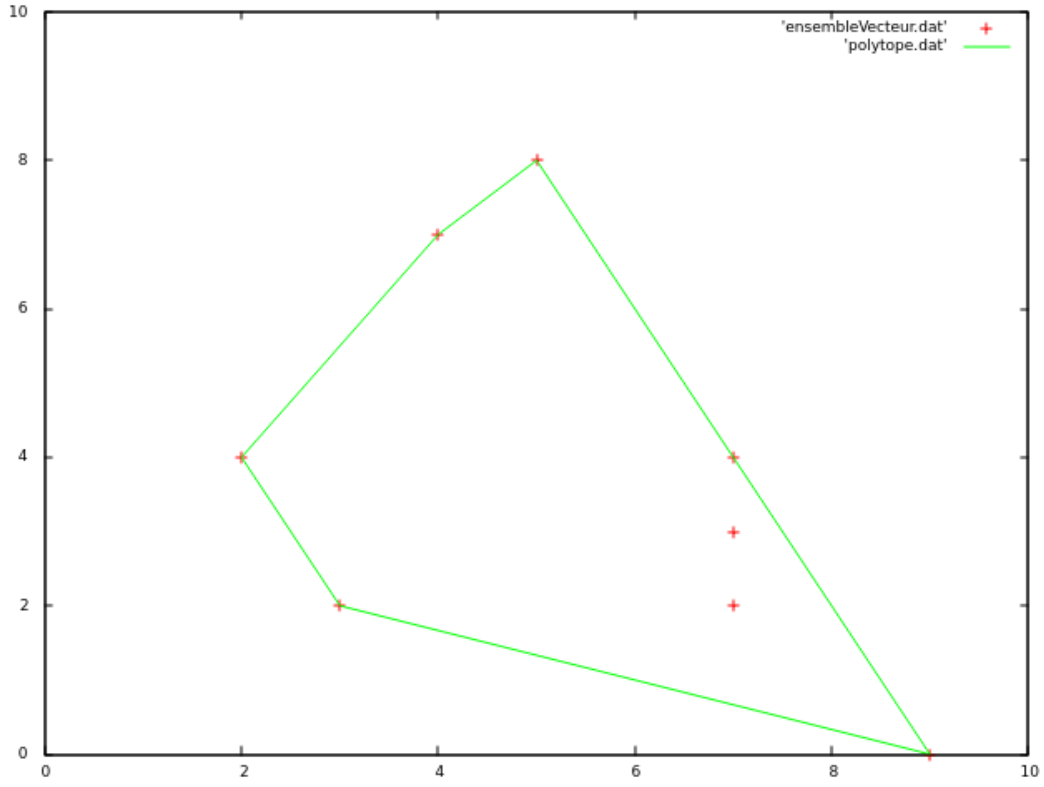

FIGURE 4.1 – Un polygone généré dans un carré de côté $k = 8$ en calculant l'enveloppe convexe de 8 points

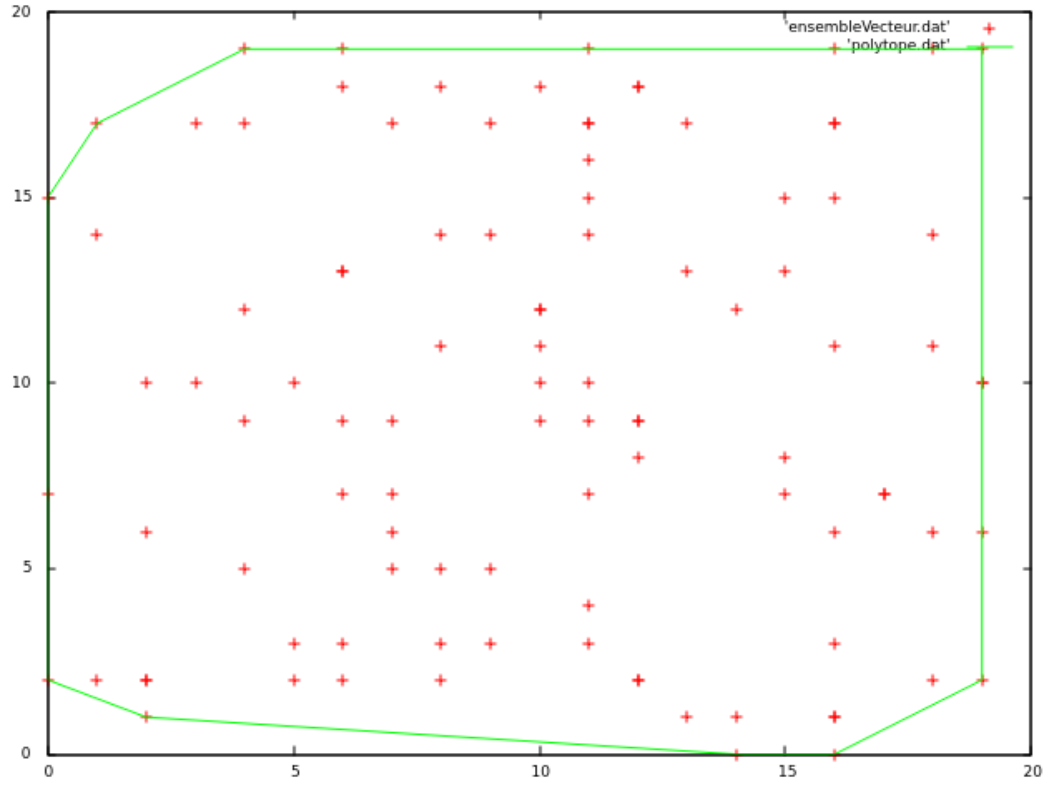

FIGURE 4.2 – Un polygone généré dans un carré de côté $k = 20$ en calculant l'enveloppe convexe de 100 points

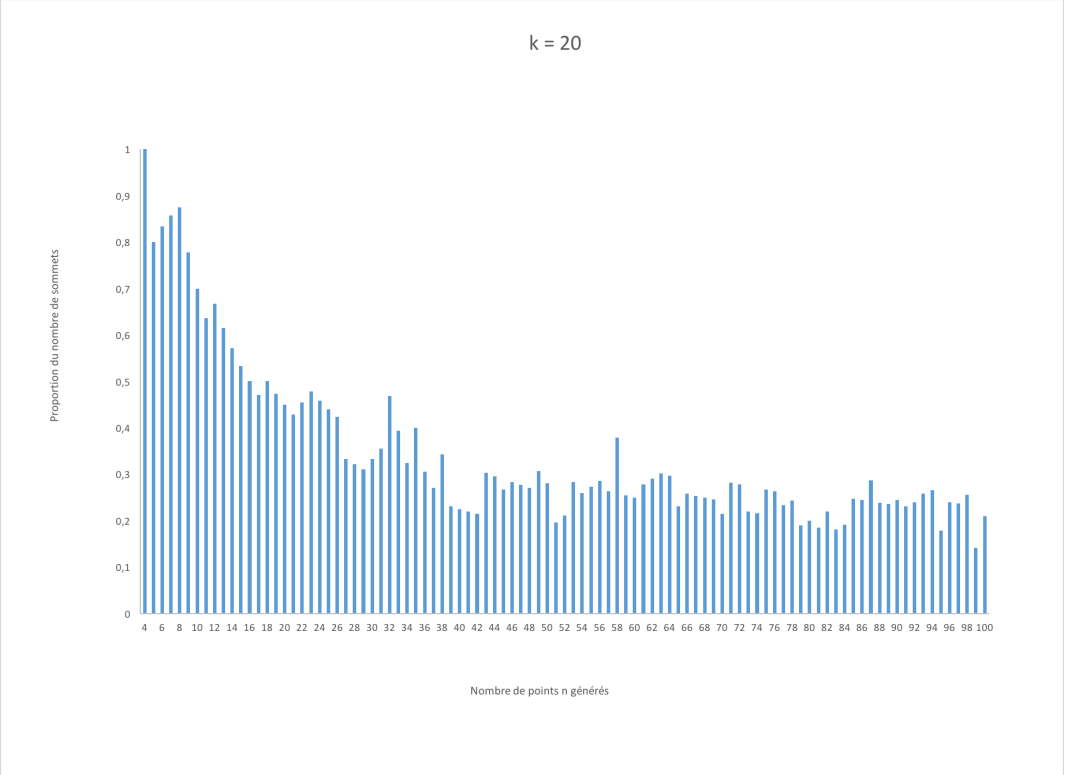

FIGURE 4.3 – Proportion du nombre de sommets du polytope obtenu par rapport au nombre n de points générés

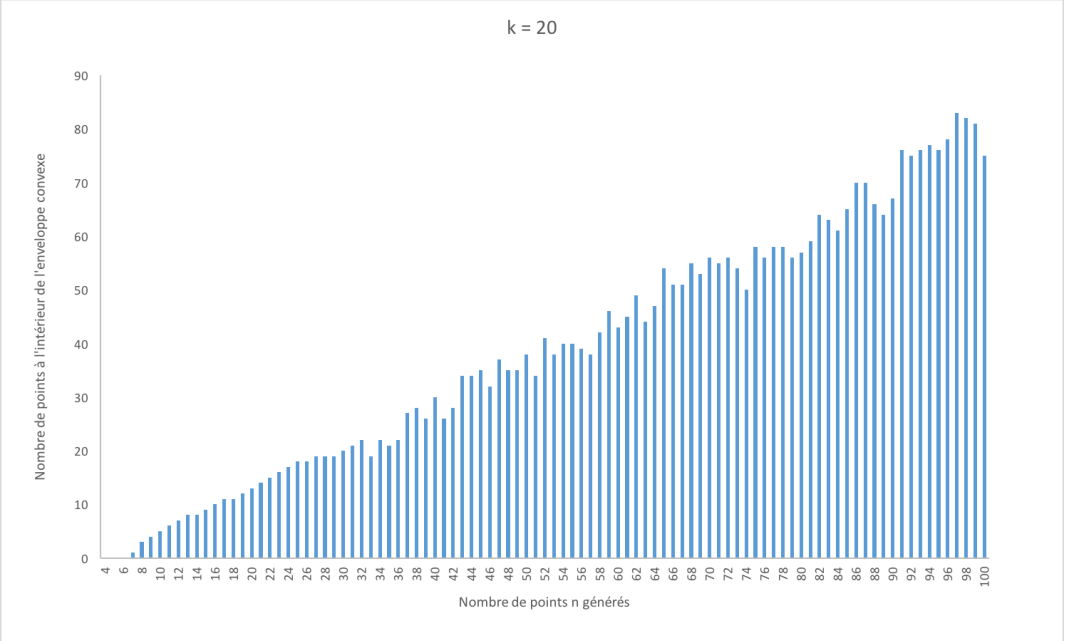

FIGURE 4.4 – Nombre de points générés à l'intérieur de l'enveloppe convexe pour un nombre n de points générés

4.4 Conclusion

Malgré le fait que l'on obtienne un polygone convexe aléatoire d'une manière assez simple, le plus grand inconvénient de cette méthode naïve est que l'on ne puisse pas paramétrer le nombre de sommets du polygone à la sortie. Cependant, de par les résultats on peut conclure que plus le nombre de points n tirés augmente, plus le nombre de points qui tombent à l'intérieur de l'enveloppe convexe est grand. En d'autres termes, beaucoup trop de points ont été générés inutilement. L'idéal serait de tirer ces n points de manière à ce qu'ils soient directement les sommets du polytope engendré. De cette façon, on aura $n = v$. De là est venue l'idée d' engendrer directement les sommets du polygone.

Chapitre 5

Génération exhaustive

Cette section présente l'essentiel du travail effectué dans la génération aléatoire de polygone convexe dont une partie détaillera la construction du générateur exhaustif mentionné dans la section 2.1.3. Toutes les étapes du développement du générateur y sont détaillées.

Le principe du générateur est le suivant : un polygone est défini comme étant une suite de quatre chemins convexes monotones. Engendrer un polygone convexe à v sommets dans un carré revient donc à tirer ces quatre chemins convexes de telle sorte que la somme de leurs sommets soit v et les relier ensuite. On s'est proposé de calculer le nombre de chemins convexes pour un nombre de sommets fixé, pour le côté k fixé de notre carré. De cette manière, on obtient le nombre de polygone pour chaque nombre de sommets. Ce nombre de sommets du polygone va définir la fonction de taille pour la classe combinatoire des polygones convexes inscrits dans le carré de côté k. Ainsi, la série génératrice sera définie pour k fixé puisqu'on aura le nombre d'éléments de chaque taille [voir section 2.1.2]. De plus, cette énumération va constituer notre générateur exhaustif.

5.1 Description de la méthode

Le plus gros du travail s'est donc basé sur le comptage du nombre de polygones convexes de chaque taille pour k fixé. Et l'on a procédé comme suit :

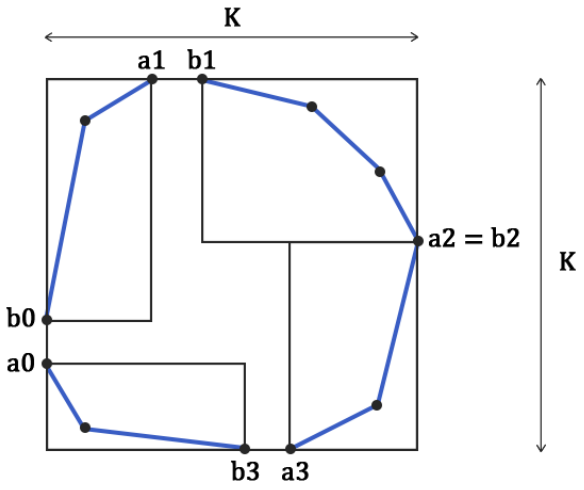

FIGURE 5.1 – Description de l'algorithme

1. Choisir quatre points a_i (à valeurs entières comprises dans $[0, k-1]$), i allant de 0 à 3 et les placer sur chaque côté du carré.
2. Choisir quatre points b_i, i allant de 0 à 3 dans l'intervalle $[a_i, k-1]$.
3. Générer un chemin convexe (b_i, a_{i+1}) avec un nombre m_i de sommets de telle sorte que $\sum_{i=0}^{3} m_i = m$. Pour $i = 3$, on génère le chemin (b_3, a_0).
4. Faire ensuite une somme pour toutes les positions possibles de (a_i, b_i).

5.2 Algorithme global

Algorithm 4: Calcul du nombre de polygones convexes à m sommets

Entrée: m : nombre de sommets

Sortie: Nombre de polygones convexes

```
1 nombreDePolygones = 0
2 forall Position de a_i do
3 |   forall Position de b_i do
4 |   |   forall Répartition m = (m_0, m_1, m_2, m_3) valide do
5 |   |   |   nombreDePolygones+ = nombreChemins(a_1, b_0, m_0) ×
  |   |   |    nombreChemins(a_2, b_1, m_1) × nombreChemins(a_3, b_2, m_2)
  |   |   |    × nombreChemins(a_0, b_3, m_3)
6 |   |   end
7 |   end
8 end
9 return nombreDePolygones
```

Une fois le principe global de la méthode établie, passons maintenant à la construction détaillée de l'algorithme. En effet, tout est basé sur la manière de compter le nombre de chemins convexes entre les points b_i et a_{i+1} placés sur les côtés du carré.

5.3 Compositions d'entiers

Soient $n \in \mathbb{N}$ et $k \in \mathbb{N}$ tel que $n \geq k$.

Définition 5.1. *Une composition de n en k parts est une séquence* $(x_1, x_2, ...x_k)$ *où* $x_i \in \mathbb{N}_+, \forall i$ *allant de 1 à k telle que*

$$\sum_{i=1}^{k} x_i = n.$$

Les x_i *sont appelés sommants ou parts, tandis que n sera la taille de la composition.*

En représentant les sommants par des petites boules ("•") unaires, on

peut représenter graphiquement une composition d'entiers en plaçant des séparateurs entre les certaines boules.

Exemple 5.1. $1+3+1+2+1=8$ *est une composition de 8 en 5 parts.*

$$1+3+1+2+1=8 \equiv \bullet|\bullet\bullet\bullet|\bullet|\bullet\bullet|\bullet$$

Définition 5.2. *L'ensemble de toutes les compositions de n en k parts est l'ensemble* $\mathcal{C}(n,k)$ *tel que :*

$\mathcal{C}(n,k) = \{(x_1, x_2, ...x_k) | (x_1, x_2, ...x_k)$ *soit une composition de n en k parts*$\}$

Proposition. *Le nombre de toutes les compositions de n en k parts est*

$$\binom{n-1}{k-1}$$

En considérant cette représentation graphique, trouver le nombre de toutes les compositions de n en k part revient à trouver toutes le manières possibles de placer $k-1$ séparateurs dans $n-1$ espaces. D'où $\text{Card}(\mathcal{C}(n,k)) = \binom{n-1}{k-1}$

Exemple 5.2. *Les* $\binom{4}{2} = 6$ *compositions de 5 en 3 parts, éléments de* $\mathcal{C}(5,3)$*, sont :*

$$\{(3,1,1),(2,2,1),(2,1,2),(1,3,1),(1,2,2),(1,1,3)\}$$

On remarque que pour une composition d'entiers, l'ordre des sommants compte. En effet, on peut voir sur l'exemple ci-dessus que $3+1+1=5$ est une composition de 5 en 3 parts différente de $1+1+3=5$.

5.3.1 Ecriture binaire

Soit c_1 un élément de $\mathcal{C}(n,k)$. c_1 peut être représente par un mot binaire dans lequel on écrit 1 si il y a un séparateur et 0 sinon. Cette écriture est la retranscription binaire de la représentation graphique d'une composition d'entiers.

Exemple 5.3. *Reprenons cette composition de 8 en 5 parts,*
$1+3+1+2+1=8$

$$\bullet|\bullet\bullet\bullet|\bullet|\bullet\bullet|\bullet \quad \equiv 1001101$$

L'écriture binaire d'une composition de n en k parts est un mot binaire de taille $(n-1)$ formé de $(k-1)$ 1 et de $(n-k)$ 0. Donc, énumérer tous les éléments de $\mathcal{C}(n,k)$ revient à énumérer tous les mots binaires de taille $n-1$ dans lequel 1 apparaît exactement $k-1$ fois ou encore trouver le nombre de manières possibles de placer $(k-1)$ 1 dans $(n-1)$ cases.

5.3.2 Procédé de parcours des éléments de $\mathcal{C}(n,k)$

On sait qu'un élément de $\mathcal{C}(n,k)$ est un mot binaire $(n-1)$ formé de $(k-1)$ 1, le reste étant des 0. Pour trouver tous les mots binaires ayant cette forme, on procède comme suit :

1. On place les $(k-1)$ 1 tout à droite ;
2. Déplacer le 1 le plus à gauche vers la gauche et ramener tous les 1 à sa droite vers la droite ;
3. Faire de même avec le 1 suivant tant que le dernier bit du mot ne soit pas 0 ;
4. Répéter le procédé jusqu'à ce que tous les 1 soient à gauche ;

Exemple 5.4. *Pour* $\mathcal{C}(8,5)$ *on aura :*

— 0001111
— 0010111
— 0011011
— 0011101
— 0011110
— 0100111
— 0101011
— ...
— 1111000

5.3.3 De la représentation binaire au k-uplets de sommants

Dans la suite du document, il sera utile de parcourir tous les éléments de $\mathcal{C}(n,k)$. Il est plus facile de parcourir $\mathcal{C}(n,k)$ en représentant les compositions

sous forme binaires. Mais pour ce parcours, la représentation des éléments de $\mathcal{C}(n,k)$ sous forme de $k-uplets$ est nécessaire. Une manière pour transformer une représentation binaire en k-uplets de sommants est la suivante :

1. Initialiser chaque élément du k-uplet à 1, i.e. (1,1,...,1), placer un curseur sur le premier bit de la représentation binaire ;
2. Lire la représentation binaire ;
3. Si le bit sur le curseur vaut 0, ajouter 1 à l'élément du k-uplet et déplacer le curseur, sinon passer à l'élément suivant ;
4. Répéter le procédé jusqu' à ce que le curseur soit au dernier bit de la représentation binaire ;

Exemple 5.5. *Pour le premier élément en binaire de* $\mathcal{C}(8,5)$ *on aura :*

$$(0001111) \rightarrow (4,1,1,1,1).$$

5.4 Chemins polygonaux convexes

Cette section aura pour but de construire des chemins polygonaux convexes dans un rectangle de largeur n et de hauteur k avec m sommets à partir de compositions d'entiers. Et elle va reprendre les mêmes principes que la construction de polyominoes convexes de [3].

Rappelons que nous sommes dans le plan. Considérons alors deux compositions d'entiers c_1 et c_2 des éléments respectifs de $\mathcal{C}(n,m)$ et de $\mathcal{C}(k,m)$.

Définition 5.3. *Un chemin polygonal de largeur n et de hauteur k à m sommets (de longueur m) est un chemin discret défini par une paire de compositions d'entiers $c_1(n,m)$ et $c_2(k,m)$ tel que la position des sommets du chemin soit donnée par les sommants n_i et k_i de c_1 et c_2 pour i allant de 1 à m.*

Remarque. *Cette description ne prend pas en compte les chemins polygonaux ayant des pas verticaux ou horizontaux.*

Les sommants de $c_1(n,m)$ définissent les abscisses des sommets, tandis que ceux de $c_2(k,m)$ vont définir leurs ordonnées par rapport au sommet précédent.

Exemple 5.6. *Prenons* $c_1 \in \mathcal{C}(8,5)$ *et* $c_2 \in \mathcal{C}(6,5)$.

$$\left.\begin{matrix}(2,1,3,1,1)\\(2,1,1,1,1)\end{matrix}\right\} \Rightarrow \{(2,2),(1,1),(3,1),(1,1),(1,1)\}$$

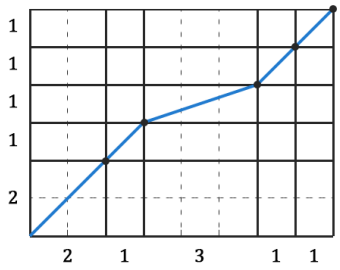

FIGURE 5.2 – Le chemin polygonal correspondant à {(2,2),(1,1),(3,1),(1,1),(1,1)}

Définition 5.4. *On définit par* $\mathcal{P}(n,k,m)$ *l'ensemble de tous les chemins polygonaux qu'on obtient par la paire de composition d'entier* $c_1(n,m)$ *et* $c_2(k,m)$.

Définition 5.5. *À chaque élément* p *de* $\mathcal{P}(n,k,m)$ *on fait correspondre un* $m-$*uplet de pentes de la manière suivante :*

$$\mathrm{p}(n,k,m) = \{p_1, p_2, ..., p_m\},\ avec\ p_i = \frac{k_i}{n_i}$$

où les n_i *et* k_i, i *allant de 1 à* m, *sont les sommants respectifs de* $c_1(n,m)$ *et de* $c_2(k,m)$.

Exemple 5.7. *Les pentes* p_i *de l'exemple précédent sont :*

$$\left.\begin{matrix}c_1 = (2,1,3,1,1)\\c_2 = (2,1,1,1,1)\end{matrix}\right\} \Rightarrow \{1,1,\frac{1}{3},1,1\}$$

Définition 5.6. *Soit* p *un élément de* $\mathcal{P}(n,k,m)$. p *est convexe si et seulement si* $\forall i$ *allant de 1 à* $m-1$, $p_i > p_{i+1}$.

Exemple 5.8. *Un chemin convexe parmi les chemins polygonaux* $\mathcal{P}(6,4,3)$

$$c_1 = (1,3,2)$$
$$c_2 = (1,2,1)$$

En décrivant ce chemin en pentes, on a $\{1, \frac{2}{3}, \frac{1}{2}\}$

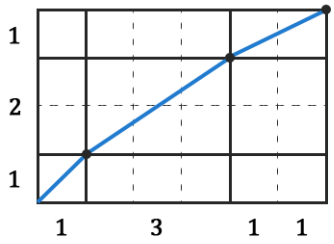

FIGURE 5.3 – Le chemin polygonal convexe correspondant à $\{1, \frac{2}{3}, \frac{1}{2}\}$

5.4.1 Nombre de chemins polygonaux convexes

Le nombre de chemins polygonaux convexes de largeur n et de hauteur k à m sommets est obtenu par l'algorithme suivant.

Algorithm 5: Calcul du nombre de chemins polygonaux convexes à m sommets

Entrée: n, k, m

Sortie: Nombre de chemins convexes

```
1  nombreDeChemins = 0
2  if m ==1 then
3  |  return 1
4  end
5  forall c1 ∈ C(n, m) do
6  |  forall c2 ∈ C(k, m) do
7  |  |  if pentesDecroissantes(c1, c2) then
8  |  |  |  nombreDeChemins++
9  |  |  end
10 |  end
11 end
12 return nombreDeChemins
```

5.4.2 Calculs préliminaires pour l'algorithme global

L'algorithme qui va calculer le nombre de polygones se base sur cette recherche du nombre de chemins. Pour des raisons de gain de temps et de

mémoire, il est préférable de faire un calcul préliminaire quant au nombre de chemins convexes pour toutes les tailles (longueur et hauteur comprises dans le carré) ainsi que tous les sommets (de 1 à v).

Algorithm 6: Calculs préliminaires

Entrée: K : côté du carré, v : nombre de sommets

Sortie: Nombre de chemins convexes

```
1  T ← NULL
2  for i = 1 to K do
3  |  T[0][i][1] = 1
4  |  T[i][0][1] = 1
5  end
6  for i = 1 to K do
7  |  for j = 1 to i do
8  |  |  for k = 1 to Minimum(i, j, v) do
9  |  |  |  T[i][j][k] = nombreChemins(i,j,k)
10 |  |  |  T[j][i][k] = T[i][j][k]
11 |  |  end
12 |  end
13 end
14 return T
```

Une fois ce prétraitement terminé, l'algorithme de comptage des polygones tirera les valeurs du tableau ainsi défini.

Remarques

1. Un sommet d'un chemin convexe est un point où la pente décroît. Définir un chemin polygonal avec une paire de compositions ne place pas seulement des sommets mais également des points sans que la pente ne décroisse. D'où la nécessité de vérifier si la pente est vraiment décroissante dans le calcul du nombre de chemins convexes.
2. Placer les quatre couples de points (a_i, b_i) sur les côtés du carré permet de gérer les cas où les pentes sont nulles. Cependant, il faut les placer

de telle manière qu'aucun pas n'ait une même pente (ici verticale ou horizontale). On procède donc de cette façon :

Choix des a_i $a_i \in [0, k-1]$

Choix des b_i si $a_{i+1} = 0$ alors $b_i = a_i$ sinon $b_i \in [a_i, k-1]$

Chapitre 6

Estimation théorique

6.1 Estimation du nombre de chemins convexes

Dans cette section, on va établir une estimation du nombre de chemins convexes de longueur m. Dans la section 5.4, on a défini un chemin convexe par une paire de compositions d'entiers. On va se baser sur ce principe pour obtenir une estimation du nombre de chemins convexes de longueur m.

Soient donc deux ensembles de compositions à m sommants $\mathcal{C}(n, m)$ et $\mathcal{C}(k, m)$. Avec ces deux ensembles, construisons tous les chemins polygonaux de largeur n et de hauteur k à m sommets.

Pour une largeur n, une hauteur k et une longueur m, on note par :

$pc(n, k, m)$ **:** Le nombre de chemins polygonaux.

$cc(n, k, m)$ **:** Le nombre de chemins polygonaux convexes.

$\bar{cc}(n, k, m)$ **:** Le nombre de chemins polygonaux non convexes.

On sait que

$$pc(n, k, m) = cc(n, k, m) + \bar{cc}(n, k, m) \tag{6.1}$$

Et que

$$pc(n, k, m) = \binom{n-1}{m-1}\binom{k-1}{m-1} \tag{6.2}$$

À partir de ces deux équations, on pourra estimer $cc(n, k, m)$. Pour ce faire, il est plus facile de trouver une expression de $\bar{cc}(n, k, m)$ et d'en déduire

$cc(n,k,m)$ par la suite.

Rappelons qu'un chemin de longueur m est convexe si et seulement si $\forall i$, allant de 1 à $m-1$, $p_i > p_{i+1}$ [voir la définition 5.6 du chapitre 5]. Alors, un chemin de longueur m n'est pas convexe si $\exists i \in [1, m-1]$ tel que $p_i \leq p_{i+1}$.

Posons $\bar{cc}(n,k,m,i)$ le nombre de chemins non convexes tel que i soit la plus petite valeur qui vérifie $p_i \leq p_{i+1}$.

On a alors :

$$\bar{cc}(n,k,m) = \sum_{i=1}^{m-1} \bar{cc}(n,k,m,i) \tag{6.3}$$

Ensuite, pour une description en pentes d'un chemin non convexe, on peut décomposer le chemin de cette manière (i étant la valeur définie précédemment) :

$$\{\underbrace{\frac{k_1}{n_1}, \frac{k_2}{n_2}, \frac{k_3}{n_3}, ...,}_{i-1 \text{ paquets}} \overbrace{\frac{k_i}{n_i}, \frac{k_i+1}{n_i+1}}^{2 \text{ paquets}}, \underbrace{..., \frac{k_m-1}{n_m-1}, \frac{k_m}{n_m}}_{m-i-1 \text{ paquets}}\}$$

Cette décomposition nous donne :

— $i-1$ pentes décrivant des chemins convexes dont le nombre est - $cc(n',k',i-1)$

— 2 pentes décrivant une condition locale de non-convexité dont le nombre est $\bar{cc}(n'',k'',2)$

— $m-i-1$ pentes décrivant des chemins polygonaux dont le nombre est $pc(n-n'-n'', k-k'-k'', m-i-1)$

En sommant sur les largeurs et les hauteurs on a :

$$\begin{aligned}\bar{cc}(n,k,m,i) = & \sum_{n'=1}^{n-m-i-1} \sum_{k'=1}^{k-m-i-1} cc(n',k',i-1) \\ & \times \sum_{n''=2}^{n-n'-m+i+1} \sum_{k''=2}^{k-k'-m+i+1} \bar{cc}(n'',k'',2) \\ & \times pc(n-n'-n'', k-k'-k'', m-i-1)\end{aligned}$$

Qui donne :

$$\begin{aligned}\bar{cc}(n,k,m,i) = &\sum_{n'=1}^{n-m-i-1}\sum_{k'=1}^{k-m-i-1}\binom{n'-1}{i-2}\binom{k'-1}{i-2} - \bar{cc}(n',k',i-1)\\ &\times \sum_{n''=2}^{n-n'-m+i+1}\sum_{k''=2}^{k-k'-m+i+1} \bar{cc}(n'',k'',2)\\ &\times \binom{n-n'-n''-1}{m-i-2}\binom{k-k'-k''-1}{m-i-2}\end{aligned}$$

Enfin, en sommant sur i :

$$\begin{aligned}\bar{cc}(n,k,m) = &\sum_{i=1}^{m-1}\sum_{n'=1}^{n-m-i-1}\sum_{k'=1}^{k-m-i-1}\binom{n'-1}{i-2}\binom{k'-1}{i-2} - \bar{cc}(n',k',i-1)\\ &\times \sum_{n''=2}^{n-n'-m+i+1}\sum_{k''=2}^{k-k'-m+i+1} \bar{cc}(n'',k'',2)\\ &\times \binom{n-n'-n''-1}{m-i-2}\binom{k-k'-k''-1}{m-i-2}.\end{aligned}$$

6.2 Estimation du nombre de polygones convexes

D'après l'algorithme de la section 5.2, l'énumération des polygones à v sommets dans $[0,k]^2$ repose essentiellement sur le placement des quatre couples de points (a_i, b_i) sur chaque côté i du carré. Énumérer tous le polygones à v sommets dans $[0,k]^2$ consiste alors à parcourir toutes les positions possibles des (a_i, b_i). De ce fait, tirer aléatoirement un polygone à v sommets dans $[0,k]^2$ revient à tirer une position des (a_i, b_i) parmi toutes les positions possibles.

Comme le choix des b_i dépend de la position de a_i et celle de a_{i+1}, pour faciliter le calcul, considérons les quatre couples (a_i, b_i) comme 2 quadruplets (a_0, a_1, a_2, a_3) et (b_0, b_1, b_2, b_3).

Probabilité de tirer les quadruplets de sommets

Soient $P_{v,k}$ le nombre de polygones à v sommets dans $[0,k]^2$ et $cc(a,b,m)$ le nombre de chemins convexes de largeur a, de hauteur b et avec m sommets. La quantité $P_{v,k}$ est estimé par :

$$P_{v,k} = \sum_{a_i b_i} \prod_{i=0}^{3} \prod_{m_i} cc(a_{i+1 \bmod 4}, b_i, m_i),$$

tel que les m_i soient les sommants de la $\mathcal{C}(v, 4)$.

Lors du calcul de $P_{v,k}$, les $cc(a_{i+1 \bmod 4}, b_i, m_i)$ sont obtenus à partir de l'expression de $\bar{cc}(a_{i+1 \bmod 4}, b_i, m_i)$ établie précédemment. Il s'agit de la somme sur toutes les positions de (a_0, a_1, a_2, a_3) et de (b_0, b_1, b_2, b_3). Donc la probabilité de tirer les quadruplets de sommets est de :

$$\frac{\prod_{i=0}^{3} \prod_{m_i} cc(a_{i+1 \bmod 4}, b_i, m_i)}{P_{v,k}}, \text{ pour des } m_i \text{ fixés.}$$

Troisième partie

Expérimentations

Chapitre 7

Expérimentations et résultats

Cette partie va présenter les résultats que l'on a obtenu durant l'exécution du générateur exhaustif. Nous tenons à préciser que l'exécution s'est limitée jusqu'au côté $k = 9$ pour le carré dans lequel on génère les polygones. Le générateur prend alors deux paramètres : le côté k du carré et le nombre de sommets v du polygone.

Le générateur exhaustif engendre tous les polygones convexes à v sommets inscrits dans le carré de côté k i.e. chaque polygone engendré touche chaque côté du carré en au moins un point.

7.1 Nombre de polygones convexes

Le tableau 7.1 montre le nombre de polygones à v sommets inscriptibles dans le carré de côté k. À partir des valeurs de ce tableau, on a effectué une représentation graphique pour avoir une idée de l'évolution du nombre de polygones engendrés. Voir Figure 7.1 et Figure 7.2.

La dernière ligne du tableau 7.1 donne la somme sur les v du nombre de polygones inscriptibles dans $[0, k]^2$. La figure 7.3 donne une représentation graphique de ce nombre total de polygones pour chaque côté k. On observe une croissance exponentielle du nombre de polygones même pour une faible évolution du côté k sur la premiere courbe. La seconde courbe montre une croissance linéaire du $\log_9 N$, où N désigne le nombre de polygones.

v	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$
4	1	16	81	256	625	1296	2401	4096	6561
5	0	16	252	1600	6600	20448	53116	120832	248184
6	0	2	170	2786	20932	103424	385050	1196608	3198404
7	0	0	24	1616	25064	214408	1195752	5225136	18604408
8	0	0	1	279	11862	193029	1753673	11174686	54773092
9	0	0	0	4	1972	73464	1234372	12192112	85874892
10	0	0	0	0	84	10958	399400	6796644	72790058
11	0	0	0	0	0	540	52872	1854884	32869456
12	0	0	0	0	0	4	2319	220888	7520004
13	0	0	0	0	0	0	20	8972	784104
14	0	0	0	0	0	0	0	66	30008
15	0	0	0	0	0	0	0	0	280
16	0	0	0	0	0	0	0	0	0
Total	1	34	528	6541	67139	617571	5078975	38794924	276699451

TABLE 7.1 – Nombre de polygones à v sommets inscriptibles dans le carré de côté k.

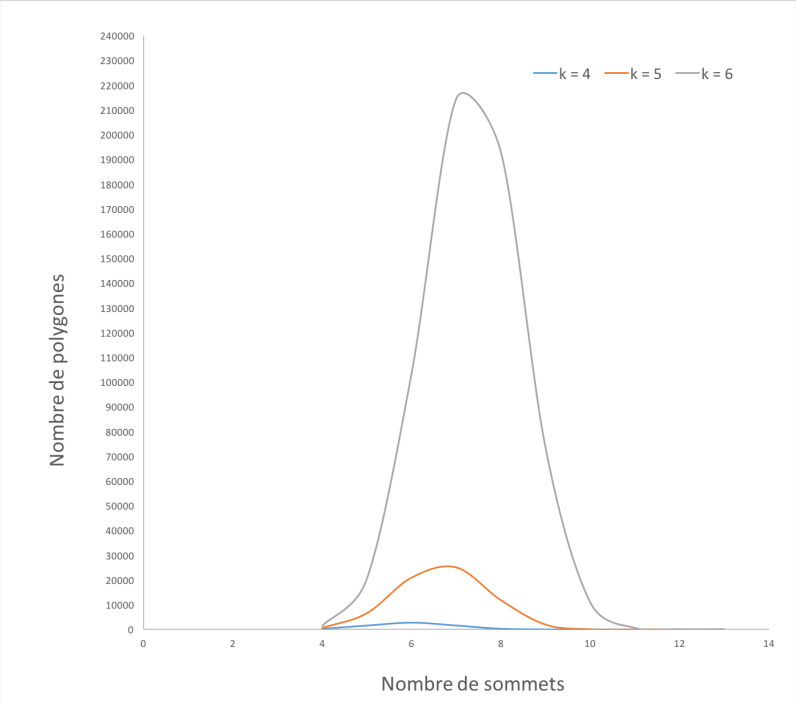

FIGURE 7.1 – Évolution du nombre de polygones suivant le nombre de sommets pour les côtés 4, 5, 6.

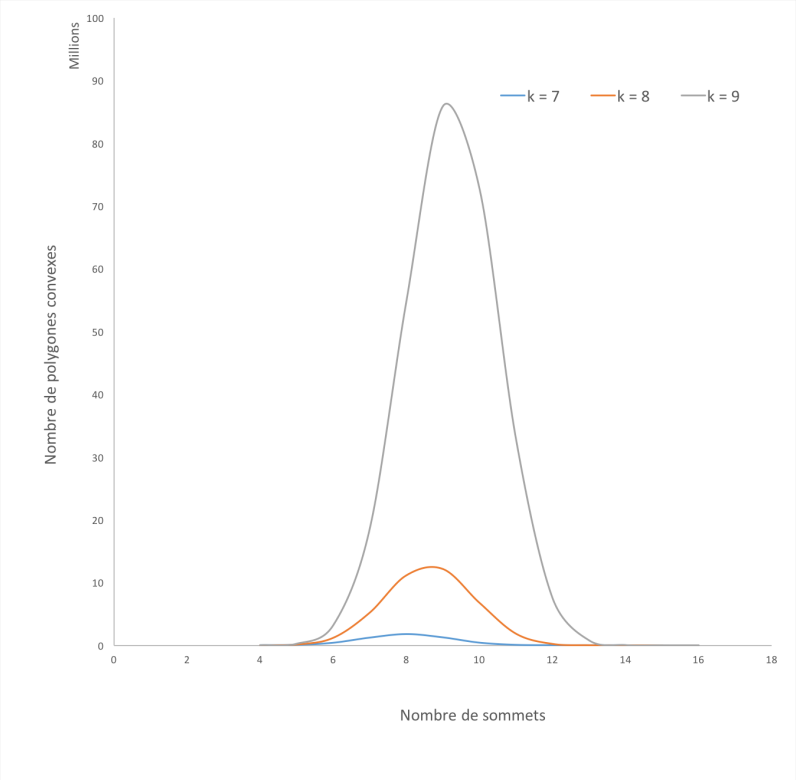

FIGURE 7.2 – Évolution du nombre de polygones suivant le nombre de sommets pour les côtés 7, 8, 9.

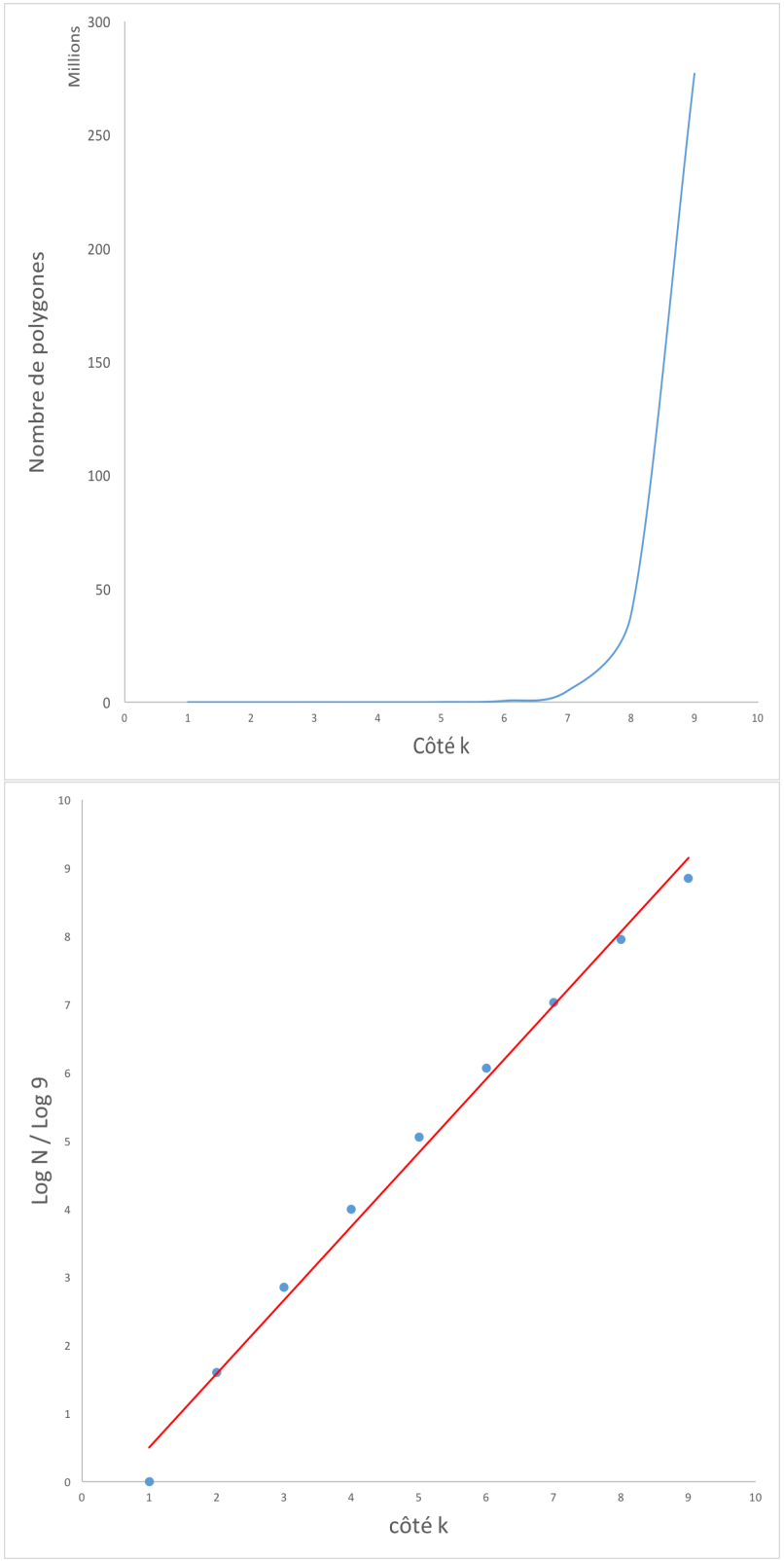

FIGURE 7.3 – Évolution du nombre de polygones par rapport au côté k

7.2 Série génératrice

Pour k fixé et en prenant comme taille le nombre v de sommets. Le tableau 7.1 donne les coefficients $[c_{k,n}]z^n$ de la série génératrice $C_k(z)$ des polygones convexes inscrits dans le carré de côté k.

Exemple 7.1. *Pour $k = 4$, on a :*

$$C_4(z) = 256z^4 + 1600z^5 + 2786z^6 + 1616z^7 + 279z^8 + 4z^9.$$

D'une manière générale cette série génératrice est une somme de la forme :

$$C_k(z) = \sum_{n=4}^{N_{max}} c_{k,n} z^n,$$

où N_{max} est le nombre tel que le carré de côté k ne puisse plus contenir aucun polygone ayant un nombre de sommets supérieur à celui-ci.

7.3 Nombre maximum de sommets et diamètre du polygone

Soit $\delta(d, k)$, le diamètre du polytope de dimension d et de côté k. Notons que ce diamètre est le diamètre au sens du graphe (il sera donc le diamètre du graphe du polytope). Comme nous sommes dans le plan, i.e. $d = 2$, on a $\delta(2, k) = \lfloor \frac{N_{max}(k)}{2} \rfloor$.

Deza-Pournin(2016). *Si $k \geq 3$ alors $\delta(d, k) \leq (k - \frac{2}{3})d$*

Le tableau 7.3 montre une comparaison des valeurs de notre générateur avec ce résultat.

k	N_{max}	$\lfloor \frac{N_{max}(k)}{2} \rfloor$	$(k - \frac{2}{3}) \times 2$
3	8	4	4,666666667
4	9	4	6,666666667
5	10	5	8,666666667
6	12	6	10,66666667
7	13	6	12,66666667
8	14	7	14,66666667
9	15	7	16,66666667

TABLE 7.2 – Diamètre du polygone comparé à l'estimation de Deza-Pournin

Conclusion

Vu les limites de la méthode naïve, le principe de génération dans [7] nous a donné l'idée de directement engendrer les points comme étant les sommets du polygone, donc de tracer des chemins convexes. Ensuite [3] nous a permis d'avoir l'idée conductrice de notre générateur de polygones dans la mesure où chaque polygone peut être assimilé à une réunion de quatre chemins convexes à pentes monotones.

La génération que nous nous sommes proposée de faire est une génération récursive qui nécessite un pré-calcul du nombre d'éléments de chaque taille. Ce pré-calcul nous a conduit à construire le générateur exhaustif qui énumère justement le nombre d'éléments de chaque taille. L'implémentation de ce générateur exhaustif représente un calcul assez lourd vu l'évolution exponentielle du nombre de polygones inscrits dans le carré de côté k. Des estimations théoriques ont également été établies en utilisant les éléments de construction du générateur exhaustif.

La génération aléatoire de polygones convexes à sommets entiers , qui sont des 2-polytopes, dans $[0, k]^2$ est un bon départ pour une bonne compréhension d'une part des polytopes, et d'autre part de la génération aléatoire. Bien que notre méthode de génération soit pesante du fait que trop de paramètres entrent en compte (les deux quadruplets de sommets), on a quand même obtenu les premières valeurs de la série génératrice des polytopes entiers dans $[0, k]^2$.

Le travail effectué a permis d'avoir des résultats intéressants en dimension $d = 2$, on a pu acquérir des éléments de réponse sur le nombre de polygones différents dans $[0, k]^d$. Néanmoins, beaucoup de questions restent encore ouvertes par exemple le comportement asymptotique de ce nombre

des polygones quand k et d tendent vers l'infini. On pourra également se poser des questions sur la forme limite des polytopes aléatoires lorsque k et d tendent vers l'infini.

En somme, le stage a constitué une étape primordiale dans l'introduction à l'étude des polytopes ainsi que de leur combinatoire. Commencer une étude en dimension 2 s'est avéré très utile pour bien s'approprier le sujet en vue d'une étude en dimension plus grande.

Bibliographie

[1] David Avis. lrslib, version 4.2. `http://cgm.cs.mcgill.ca/~avis/C/lrs.html`, 2005.

[2] Bradford Barber and Hannu Huhdanpaa. Qhull manual. *The Geometry Center, Minneapolis MN*, 2003.

[3] Olivier Bodini, Ph Duchon, Alice Jacquot, and L Mutafchiev. Asymptotic analysis and random sampling of digitally convex polyominoes. In *International Conference on Discrete Geometry for Computer Imagery*, pages 95–106. Springer, 2013.

[4] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic geometry*. Cambridge university press, 1998.

[5] Julien Bureaux and Nathanael Enriquez. Lattice convex chains in the plane. *arXiv preprint math/0612770*, 2006.

[6] Thomas Christof and Andreas Löbel. Porta–polyhedron representation transformation algorithm, 2008. 2012.

[7] Olivier Devillers, Philippe Duchon, and Rémy Thomasse. A generator of random convex polygons in a disc. In *AofA 2014-25th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms*, 2014.

[8] Jeff Erickson. Lecture on convex hulls. `http://jeffe.cs.illinois.edu/teaching/373/notes/x05-convexhull.pdf`, 2002.

[9] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. cambridge University press, 2009.

[10] Komei Fukuda. cddlib reference manual, cddlib version 0.93d. *ETHZ, Zürich, Switzerland*, 2001.

[11] Ewgenij Gawrilow and Michael Joswig. Geometric reasoning with polymake. *arXiv preprint math/0507273*, 2005.

[12] Francis Lazarus. Géométrie algorithmique notes de cours. 2014.

[13] Patrick Prosser. Convex hulls. `http://www.dcs.gla.ac.uk/~pat/52233/slides/Hull1x1.pdf`.

[14] Raimund Seidel. Linear programming and convex hulls made easy. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 211–215. ACM, 1990.

[15] Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 1995.