Simplification polyédrique optimale pour le rendu

Emilie Charrier

UNIVERSITÉ PARIS-EST
ECOLE DOCTORALE ICMS

Thèse pour obtenir le grade de docteur de l'université PARIS-EST
Spécialité : Informatique

présentée par

Émilie CHARRIER

le 4 décembre 2009

Bourse DGA (DGA/D4S/MRIS)

SIMPLIFICATION POLYÉDRIQUE OPTIMALE POUR LE RENDU

Directeur de thèse : Gilles BERTRAND
Encadrant de thèse : Lilian BUZER

Composition du jury :

Rapporteurs :
- Rémy MALGOUYRES, Professeur, Université d'Auvergne
- Valérie BERTHÉ, Directrice de Recherche CNRS, Université Montpellier II

Examinateurs :
- Jacques-Olivier LACHAUD, Professeur, Université de Savoie
- David COEURJOLLY, Chargé de recherche CNRS, Université Lyon I
- Lilian BUZER, Professeur associé, ESIEE-PARIS

Directeur de thèse :
- Gilles BERTRAND, Professeur, ESIEE-PARIS

À Julien

Cette thèse a été préparée à :

Université PARIS-EST
Équipe A3SI du Laboratoire d'Informatique de l'Institut Gaspard Monge
CNRS, UMR 8049
ESIEE-PARIS
2, bd Blaise Pascal, CITÉ DESCARTES, BP 99
93162 NOISY LE GRAND CEDEX

Grâce à une bourse de la DGA :

Délégation Générale pour l'Armement
Direction des systèmes de forces et des stratégies industrielles, technologiques et de coopération
Mission pour la recherche et l'innovation scientifique
D4S/MRIS – 00303 Armées
8, bd Victor – 75015 PARIS

Remerciements

Je tiens tout d'abord à remercier très chaleureusement Monsieur Lilian Buzer qui a été pendant ces trois années mon encadrant de thèse. Je lui suis extrêmement reconnaissante de la confiance qu'il m'a accordée depuis le premier jour. Il m'a guidée durant ma thèse tout en me laissant suffisamment d'autonomie et j'ai énormément appris à ses côtés.

J'adresse également mes remerciements les plus sincères à Gilles Bertrand qui a accepté d'être mon directeur de thèse.

J'exprime ma gratitude à tous les membres du jury qui ont bien voulu juger mes travaux de thèse. Je remercie donc vivement Monsieur Rémy Malgouyres et Madame Valérie Berthé qui ont accepté la rude charge de rapporteur. Un immense merci également à Monsieur Jacques-Olivier Lachaud et Monsieur David Coeurjolly qui ont bien voulu évaluer mon travail.

Je suis très reconnaissante envers la DGA qui m'a fait confiance en m'accordant une allocation de recherche. Ma thèse aurait été impossible sans ce financement.

Je remercie Monsieur Fabien Feschet de m'avoir proposé de collaborer avec lui durant ma troisième année de thèse, ce fut une expérience très enrichissante.

Je tiens à exprimer toute ma gratitude à l'ensemble des membres de l'équipe A3SI pour leur bonne humeur et l'accueil chaleureux qu'ils m'ont réservé à mon arrivée. Je remercie en particulier Monsieur Denis Bureau et Monsieur Michel Couprie qui m'ont donné l'opportunité d'enseigner en parallèle de ma thèse et qui, par la même occasion, m'ont formée au métier d'enseignante.

Un immense merci à tous mes collègues doctorants pour leur sympathie et leur incroyable capacité à supporter mes humeurs changeantes. Concernant ce dernier point, toutes mes félicitations à John et Yohan qui ont dû partager le même bureau que moi pendant presque trois ans !

Enfin, j'adresse toute mon affection à ma famille qui, malgré les quelques centaines de kilomètres qui nous séparent, ont toujours été d'un très grand soutien pour moi. Un énorme merci donc à ma mère, mon père, mon frère et mes grands parents ! Je remercie affectueusement mon compagnon Julien pour sa patience et son soutien au quotidien.

Résumé

En informatique, les images sont numériques et donc composées de pixels en 2D et de voxels en 3D. Dans une scène virtuelle 3D, il est impossible de manipuler directement les objets comme des ensembles de voxels en raison du trop gros volume de données. Les objets sont alors polyédrisés, c'est-à-dire remplacés par une collection de facettes. Pour ce faire, il est primordial de savoir décider si un sous-ensemble de voxels peut être transformé en une facette dans la représentation polyédrique. Ce problème est appelé reconnaissance de plans discrets. Pour le résoudre, nous mettons en place un nouvel algorithme spécialement adapté pour les ensembles de voxels denses dans une boite englobante. Notre méthode atteint une complexité quasi-linéaire dans ce cas et s'avère efficace en pratique. En parallèle, nous nous intéressons à un problème algorithmique annexe intervenant dans notre méthode de reconnaissance de plans discrets. Il s'agit de calculer les deux enveloppes convexes des points de Z^2 contenus dans un domaine vertical borné et situés de part et d'autre d'une droite quelconque. Nous proposons une méthode de complexité optimale et adaptative pour calculer ces enveloppes convexes. Nous présentons le problème de manière détournée : déterminer le nombre rationnel à dénominateur borné qui approxime au mieux un nombre réel donné. Nous établissons le lien entre ce problème numérique et son interprétation géométrique dans le plan. Enfin, nous proposons indépendamment un nouvel algorithme pour calculer l'épaisseur d'un ensemble de points dans le réseau Z^d. Notre méthode est optimale en 2D et gloutonne mais efficace en dimension supérieure.

Mots clés

Géométrie discrète, géométrie algorithmique, polyédrisation, plan discret, enveloppe convexe, théorie des nombres, grilles (analyse numérique)

Title

Optimal Polyhedral Simplification for Rendering

Abstract

In computer science, pictures are digital and so, they are composed of pixels in 2D or of voxels in 3D. In 3D virtual scenes, we cannot directly manipulate objects as sets of voxels because the data are too huge. As a result, the objects are transformed into polyhedra, i.e. collections of facets. For this, we must be able to decide if a subset of voxels can be replaced by a facet in the polyhedrisation. This problem is called digital plane recognition. To solve it, we design a new algorithm especially adapted for sets of voxels which are dense in a bounding box. Our method achieves a quasi-linear worst-case time complexity in this case and it is efficient in practice. In parallel, we study another algorithmic problem which occures in our digital plane recognition algorithm. It is computing the two convex hulls of grid points lying in a bounded vertical domain and located on either side of a straight line. We propose an optimal time complexity method to compute these convex hulls and which is also output sensitive. We present the problem in a different way : find the rational number of bounded denominator that best approximates a given real number. We establish the link between this numerical problem and geometry. Finally, we independently propose a new algorithm to compute the lattice width of a set of points in Z^d. Our method is optimal in 2D and is greedy but efficent in higher dimension.

Keywords

Digital geometry, computational geometry, polyhedrisation, digital plane, convex hull, number theory, lattice

Table des matières

Table des figures

Liste des Algorithmes

Introduction

Un Petit Morceau de ma Vie

À mon entrée à l'université, si quelqu'un m'avait dit qu'en 2009 je soutiendrais ma thèse de doctorat en informatique, je ne l'aurais probablement pas cru. À l'époque, j'ai une vision assez réduite de l'informatique car je la connais mal. Mon baccalauréat scientifique en poche, je me lance alors dans des études universitaires sans trop en connaître la finalité. Un conseiller d'orientation m'indique un DEUG en Mathématiques et Informatique Appliqués aux Sciences... Je me lance, même si le mot "informatique" me fait un peu peur. Après tout, j'aime bien les mathématiques. De plus, ce qui me motive, c'est la perspective de pouvoir devenir enseignante et ça ne me semble pas incompatible. Très vite, les enseignements en informatique m'ont interpellée, en particulier l'algorithmique, probablement en grande partie grâce à des enseignants très motivés et convaincants. Je choisis donc de poursuivre mes études en informatique avec une licence et une maîtrise.

De fil en aiguille, je m'essaie à la recherche scientifique et je découvre plus précisément le métier d'enseignant chercheur. C'est exactement ce que je veux devenir, j'ai enfin trouvé ma voie. Même si mes enseignants me mettent en garde de la difficulté à pratiquer la recherche en France de nos jours, je me lance dans un master recherche en *Informatique Fondamentale* et je découvre lors d'un cours d'option la géométrie discrète. J'effectue alors mon stage de recherche dans ce domaine et prends définitivement goût à cette discipline. Ne parvenant pas dès la fin de mon master à décrocher une bourse pour une thèse dans ce domaine, j'enchaîne avec un second master en *Imagerie, Vision et Robotique*. Ce que je veux faire, c'est une thèse en géométrie discrète et si ce n'est pas cette année, ce sera l'année prochaine. Il n'est pas question de revoir maintenant mes objectifs. En 2006, je décroche enfin une bourse de thèse pour effectuer des recherches en géométrie discrète et algorithmique. C'est en région parisienne, mais ça vaut le coup.

La Géométrie... Discrète

Quand on me demande quel est mon domaine de recherche, je réponds que je travaille en géométrie discrète. Les gens me rétorquent souvent : "De la géométrie ? Mais alors tu fais des maths, je croyais que tu faisais de l'informatique !". Eh bien justement, la géométrie discrète résulte de la rencontre de la géométrie du continu avec l'informatique ou plutôt avec le monde numérique. Mais avant de définir plus précisément la géométrie discrète, revenons quelques instants sur les origines de la géométrie classique.

Dès le paléolithique, l'homme préhistorique crée des formes géométriques dans ses peintures et gravures, sur des supports plats et cylindriques comme les murs des grottes ou les os d'animaux. Il taille également les pierres et façonne ainsi des outils mettant en oeuvre la notion de symétrie par exemple. Cependant, le mot "géométrie" n'apparaît

[ETRES]	DISCRETS divisés	CONTINU unifiés
INDETERMINEE	multitude	grandeur
DETERMINEE	en soi / en relation	non−mu / mu
SCIENCES CORRESPONDANTES	arithmétique musique	géométrie sphérique

FIG. 1 – Système de Nicomaque de Gérase

que beaucoup plus tard. Dans la grande encyclopédie de Diderot et D'Alembert (1751-1772), on trouve : "*La géométrie est la science des propriétés de l'étendue(...). Ce mot est formé de deux mots grecs, γϵ ou γαια, terre, et μϵτρια, mesure ; et cette étymologie semble nous indiquer ce qui a donné naissance à la géométrie : imparfaite et obscure dans son origine comme toutes les autres sciences, elle a commencé par une espèce de tâtonnement, par des mesures et des opérations grossières, et s'est élevée peu à peu à ce degré d'exactitude et de sublimité où nous la voyons.*". Le mot *géométrie* apparaît pour la première fois dans un livre d'Hérodote vers 445 avant JC. Selon lui, la géométrie est originaire d'Égypte. Le roi Sésostris aurait décidé de partager les terres en parcelles et de distribuer un lot à chaque habitant en échange d'une redevance annuelle. Les crues du Nil recouvrant parfois une partie des parcelles, les terrains restants auraient été mesurés afin de diminuer proportionnellement l'impôt. La géométrie serait donc bien liée à l'origine à la mesure des terres. Cependant, la Grèce antique est plutôt considérée comme le berceau de la géométrie. Les Grecs auraient ramené ce savoir chez eux. Il s'agirait peut être de Thalès dont la légende raconte qu'il aurait impressionné le roi Amasis en mesurant les pyramides égyptiennes grâce à leur ombre. Malheureusement, la plupart des écrits de cette époque sont perdus. Nous savons que la géométrie est enseignée dès le V^{e} siècle à Athènes et constitue alors une discipline des mathématiques à part entière. Platon organise un système mathématique selon quatre spécialités : l'arithmétique, la géométrie, l'astronomie et l'harmonique (théorie mathématique des intervalles musicaux). Par la suite, Nicomaque de Gérase tente de justifier ce système par la philosophie en distinguant le discret du continu, l'en-soi et le relatif ainsi que l'immobile et le mouvement (voir la figure 1). Nous remarquons que dans ce système la géométrie est uniquement associée au continu. Nicomaque de Gérase aurait-il imaginé que cela puisse être autrement dix-huit siècles plus tard ?

Le premier texte géométrique connu est *Éléments* d'Euclide au III^{e} siècle avant JC (voir la figure 2). Il comporte treize livres principalement consacrés à la géométrie mais incluant également trois livres d'arithmétique traitant de la théorie des nombres. Ces livres sont à l'origine de la géométrie euclidienne bien connue de nos jours.

Faisons à présent un grand, voire un énorme, saut dans le temps qui nous amène aux années 1960-1970, époque de l'émergence de la géométrie discrète sous l'impulsion du développement de l'informatique. Les images manipulées par ordinateurs sont dites *numériques* et sont constituées de pixels en dimension 2 ou de voxels en dimension 3. Nous reviendrons plus en détail sur ces notions dans la section 1.1.1. Considérons des images en 2D par exemple. Les pixels sont disposés de manière régulière de façon à couvrir le

FIG. 2 – Les Éléments d'Euclide

support permettant d'afficher les images, à savoir l'écran de l'ordinateur. Ce support est appelé espace discret. Les pixels peuvent être assimilés à des diodes lumineuses et ainsi un pixel allumé appartient à l'objet discret et un pixel éteint ne lui appartient pas. Dans les images numériques, les objets géométriques que nous connaissons dans le continu n'ont plus vraiment la même allure et surtout, ils n'ont plus les mêmes propriétés. Par exemple, un segment de droite dans le continu correspond à une infinité de points à coordonnées réelles. Dans l'espace discret en revanche, il correspond à un nombre fini de pixels allumés dessinant un morceau de courbe "plus ou moins droit" appelé segment de droite discrète. De plus, les nombres réels n'existent pas dans un espace discrets et les pixels sont exprimés avec des coordonnées entières. La géométrie discrète a donc été créée pour caractériser les nouveaux objets géométriques dans l'espace discrets et être capable de les manipuler grâce à des algorithmes adaptés. Nous devons les fondements de la géométrie discrète à Azriel Rosenfeld au début des années 1970 (voir par exemple [Ros70]). La géométrie discrète est une nouvelle branche des mathématiques et l'étude des structures discrètes relèvent de la combinatoire ou encore de l'arithmétique. Donc, lorsqu'on me demande si je fais des mathématiques ou de l'informatique, je réponds : "Un peu des deux !". Ceci n'éclaire pas toujours mes interlocuteurs mais pourtant c'est un peu ça, la géométrie discrète.

Présentation du Sujet de Thèse

Cette thèse de doctorat a été rendue possible grâce à une bourse de la DGA (Délégation Générale pour l'Armement) et a pour titre "Simplification polyédrique optimale pour le rendu". De nos jours, l'ensemble des périphériques de réalité virtuelle utilisent un modèle polyédrique. Ainsi, la surface d'un objet tri-dimensionnel est décrite comme une collection de facettes triangulaires représentant la forme globale de l'objet. Par la suite, des textures sont plaquées sur ces facettes afin de synthétiser l'image finale. Cette étape est communément appelée le *rendu*. L'information géométrique de l'objet doit être représentée avec un maximum de vraissemblance afin de favoriser l'effet d'immersion dans le monde virtuel et éviter des images trop artificielles. Selon l'éloignement de l'objet dans la scène virtuelle, le polyèdre associé comportera plus ou moins de facettes. Par exemple, pour un objet positionné au premier plan, nous utiliserons une description de l'ordre de mille facettes contre une cinquantaine seulement pour un objet vu d'une dizaine mètres.

Techniquement, les méthodes de simplification actuelles utilisent un critère d'approximation mathématique à partir d'une description des objets en surfaces complexes telles que les Nurbs. Ces approches fournissent des résultats mathématiquement valides mais ne fournissent aucun contrôle sur le rendu des objets et produisent donc des résultats visuellement approximatifs. Lilian Buzer a initié ces dernières années une thématique de recherche sur la simplification géométrique spécifiquement adaptée aux périphériques discrets. Se basant sur le fait que la précision du rendu est intrinsèquement limitée par celle du périphérique d'affichage, L. Buzer fournit en 2002 le premier algorithme permettant de simplifier des objets polygonaux suivant un écart maximal d'un demi-pixel [Buz02]. Il obtient ainsi un algorithme quasi-linéaire alors que les approches existantes étaient au mieux quadratiques.

La réalité virtuelle a une importance majeure pour la préparation à la défense puisqu'elle est utilisée dans le cadre de simulations de tirs de missiles, par exemple. L'évaluation de la chaîne de guidage terminal des missiles tactiques est indispensable pour la maîtrise des risques techniques sur les programmes de missiles de croisière (SCALP-EG, MDCN), sol-air (MISTRAL RMV) ainsi que pour la préparation des futurs défis technologiques (DAMB, Missile de Combat Terrestre, autodirecteurs multispectraux). Le nombre de tirs de missiles étant réduit désormais dans les programmes pour des raisons économiques, une orientation forte vers la qualification par simulation a été prise par la DGA. La simulation hydride qui permet de vérifier les performances des autodirecteurs infrarouges en laboratoire nécessite l'utilisation de moyens de projection d'images IR de synthèse des cibles dans leur environnement. La représentativité de ces images vis à vis de la réalité doit encore être améliorée pour garantir la précision des simulations et in fine la qualité des résultats d'évaluation des performances des autodirecteurs. Le premier objectif de cette thèse était d'étendre les résultats obtenus par Lilian Buzer à la dimension 3 en fournissant des méthodes de complexité optimale. Par ailleurs, le but était d'implémenter ces algorithmes dans les logiciels utilisés par le service de la DGA concerné.

Présentation des Travaux

Les images tri-dimensionnelle dont nous disposons sont généralement acquises par capteurs numériques et correspondent à un ensemble de voxels (équivalent du pixel en 3D). Les voxels sont assimilables à des points à coordonnées entières dans l'espace tri-dimensionnel, disposés sur une grille régulière. Avant la mise en place d'une méthode finale de polyédrisation ou de simplification polyédrique à partir de ces images, il convient de posséder au préalable un algorithme efficace de reconnaissance de plans discrets. Cette problématique consiste à déterminer si un ensemble de points dans $\mathbb{Z}^3$, également appelé ensemble de voxels, correspond à la discrétisation d'un morceau de plan euclidien. La reconnaissance de plans discrets est donc utilisée pour décider si une partie d'un objet discret est plate, et par extension, pour polyédriser un objet discret. En effet, si un sous-ensemble de voxels correspond à une partie plate de l'objet, alors il pourra être remplacé par une facette dans la polyédrisation finale. De plus, il est possible de relaxer le critère de planitude en considérant des morceaux de plans discrets plus ou moins épais en vue de polyédrisation plus grossières. La reconnaissance de plans discrets est un problème largement étudié en géométrie discrète (voir [BCK07]). Ces approches sont généralement en relation avec la programmation linéaire [Buz03, Meg84, ST91, BD07], les enveloppes convexes et la géométrie algorithmique [DR95, DRR94, PS85, VC00, Kim84, KS91, ST91], l'optimisation combinatoire [DR95, DRR94, FST96, GDRZ05, Rev95, VC00] ou encore la propriété de régularité [Vee94, Vee93]. La plupart des méthodes existantes ne combinent pas à la fois une complexité intéressante dans le pire cas et une efficacité en pratique. En 2006, L. Buzer propose une nouvelle méthode [Buz06] atteignant une complexité dans le pire cas en $O(n \log^2 D)$ où $D-1$ représente la plus grande longueur d'une boite englobant l'ensemble de points. De plus, sa méthode reconnaît un ensemble dense de 10^6 points en environ 360 itérations linéaires. La méthode de Buzer transforme le problème de reconnaissance de plans discrets naïfs en un problème de réalisabilité sur une fonction convexe bi-dimensionelle dite *fonction épaisseur*. Par conséquent, cette méthode ne prend en compte que deux paramètres et elle utilise des techniques géométriques planaires afin de déterminer si l'espace des solutions 2D est vide. Pour résoudre le problème de réalisabilité, elle combine l'oracle de Megiddo et une recherche dichotomique mono-dimensionnelle. Nous nous sommes proposés de reprendre cette méthode et d'en améliorer à la fois la complexité dans le pire cas et l'efficacité en pratique. Le principal changement apporté consiste en l'utilisation d'une propriété du centre de gravité durant la résolution du problème de réalisabilité. Ceci ramène la complexité globale de la méthode à $O(n \log D)$. De plus, des améliorations pratiques nous permettent de réduire le nombre d'itérations de la méthode à seulement une dizaine pour un ensemble de 10^6 points. Ce travail a donné lieu à une première publication scientifique [CB08a], une seconde étant en cours d'élaboration.

Comme nous l'avons évoqué, la méthode de reconnaissance de plans discrets proposée résoud un problème de réalisabilité sur une fonction convexe bi-dimensionnelle appelée *fonction épaisseur* et noté *ep*. Nous cherchons à déterminer un point u du domaine de recherche tel que $ep(u)$ est strictement inférieure à 1. Pour cela, un domaine de recherche bi-dimensionnel est initialisé comme un rectangle sur une grille régulière. Le pas de discrétisation est choisi de façon à s'assurer que si le problème est réalisable, alors au moins un point à coordonnées entières de l'espace de recherche discrétisé est solution du problème. À l'initialisation, le domaine de recherche correspond donc à un polygone convexe à sommets portés par la grille. Par la suite, le centre de gravité du domaine de recherche est

calculé et la fonction épaisseur est évaluée en ce point. Si la fonction épaisseur en ce point est strictement inférieure à 1, alors le problème est réalisable et l'algorithme se termine. Si ce n'est pas le cas, l'algorithme élimine une partie du domaine de recherche. Pour cela, il le coupe par une droite passant par le centre du gravité. Le domaine de recherche en résultant correspond donc toujours à un polygone convexe, mais ses sommets correspondent alors à des points à coordonnées rationnelles dont le numérateur et le dénominateur ne cesseront de croître au fil des itérations. Pour assurer l'efficacité de la méthode, il est indispensable de conserver à chaque itération un domaine de recherche convexe à sommets entiers. Pour cela, il est nécessaire de savoir calculer l'enveloppe convexe des points de la grille situés à l'intérieur de domaine de recherche après chaque coupe. En réalité, cette enveloppe convexe n'est à reconstruire que le long de trois côtés du domaine de recherche : le côté porté par la droite de coupe et les deux côtés qui lui sont adjacents. Nous avons donc recherché dans la littérature des algorithmes calculant une telle enveloppe convexe. Malheureusement, peu de recherches ont été menées dans ce domaine. Nous sommes finalement parvenu à trouver un algorithme proposé par Harvey dans [Har99] qui utilise la théorie des nombres et en particulier l'interprétation géométrique de la décomposition en fractions continues de la pente de la droite de coupe. Cette méthode a une complexité logarithmique en la taille des coefficients des droites portant les côtés du domaine de recherche. Cependant, cette méthode n'est pas évidente à programmer. Par ailleurs, devant le peu de résultats trouvés dans la littérature, nous avons eu envie de mener nos propres recherches concernant cette problématique, celle-ci nous paraissant importante dans le domaine de la géométrie discrète et algorithmique. Pour cela, nous avons considéré le problème de façon un peu différente.

Nous considérons le problème suivant : déterminer le nombre rationnel à dénominateur borné qui approxime au mieux un nombre réel a donné. Plus formellement, il s'agit de trouver le nombre rationnel de la forme p/q qui approxime au mieux le réel a tel que son dénominateur q appartienne à l'intervalle entier borné $[b, b']$. Le problème géométrique correspondant consiste à déterminer le point à coordonnées entières situés dans un domaine vertical D de la forme $\{(x, y) \in \mathbb{R}^2 | b \leq x \leq b'\}$ tel que la droite passant par l'origine et par ce point approxime au mieux la droite L de pente a passant par l'origine. Nous remarquons que le point associé à la meilleure approximation correspond à un sommet des enveloppes convexes des points de la grille situés au-dessus ou en dessous de la droite L et appartenant au domaine vertical D. Par conséquent, si nous connaissons les sommets de ces deux enveloppes, nous pouvons facilement déterminer la droite approximante que nous recherchons. Pour calculer ces deux enveloppes convexes, nous avons trouvé dans la littérature deux algorithmes. La première méthode est celle d'Harvey et a une complexité en $O(\log(b'))$. Cependant, cette complexité peut être améliorée surtout si b' est beaucoup plus grand que la largeur du domaine D. La seconde méthode utilise la géométrie algorithmique comme l'algorithme proposé par Balza-Gomez et al. dans [BGMM99] qui atteint une complexité en temps de $O(\log(b' - b))$. Nous proposons une nouvelle méthode pour calculer ces deux enveloppes convexes. Notre méthode combine à la fois la théorie des nombres et la géométrie algorithmique. Notre algorithme préserve la complexité optimale en temps de la méthode de Balza-Gomez. Notons que notre méthode est la première dont la complexité dépend également de la taille de la sortie. Un tel algorithme est qualifié de *output sensitive* ou algorithme adaptatif. En effet, notre méthode calcule un sommet d'une des deux enveloppes convexes à chaque itération. Les itérations s'exécutant en temps constant, notre méthode a une complexité en $O(h)$ où h représente le nombre de sommets

des deux enveloppes convexes calculées. De plus, l'algorithme résultant est plus simple et s'avère donc plus adapté pour l'implémentation. Notre méthode fonctionne également lorsque la droite L ne passe pas par l'origine et lorsque le domaine est borné par des droites non-verticales. Par conséquent, elle s'adapte parfaitement lors de la reconstruction du domaine de recherche pour le problème de reconnaissance de plans discrets. Ce travail a donné lieu à trois publications scientifiques [CB08c, CB08b, CB09].

Lors de la troisième année de thèse, nous avons été invités par Fabien FESCHET, professeur au laboratoire LAIC (Laboratoire d'Algorithmique et Image de Clermont), à collaborer avec lui. F. Feschet propose dans [Fes06] un premier algorithme planaire pour le calcul de l'épaisseur d'un ensemble de points dans un réseau avec une interprétation géométrique. Sa méthode s'exécute en $O(n \log n)$ avec n le nombre de points de l'ensemble considéré. Par la suite, F. Feschet a eu l'idée d'une nouvelle méthode planaire s'exécutant en temps linéaire suivant le nombre de points de l'ensemble. Cependant, aucune publication n'avait encore vu le jour et le but de cette collaboration était de formaliser de manière précise sa méthode et d'étendre ces résultats en toute dimension. Nos travaux communs nous ont permis de formaliser de manière précise la méthode proposée en toute dimension. De plus, l'algorithme résultat est linéaire suivant le nombre de points de l'ensemble en 2D, cette complexité étant optimale. Dans le cas où la dimension est quelconque, nous avons proposé une méthode gloutonne qui s'avère efficace en pratique. Ce travail a donné lieu à une publications scientifiques [CBF09].

Le lecteur remarquera que les travaux menés durant cette thèse sont assez variés mais ne remplissent pas tous les objectifs établis dans le sujet de thèse DGA au début de l'étude. Cependant, les résultats proposés ont donné lieu à plusieurs publications ce qui confirme leur valeur scientifique dans les domaines de la géométrie discète et de la géométrie algorithmique. De plus, suite à des réunions en présence des suiveurs de la DGA, nous nous sommes aperçus qu'il était très difficile d'implémenter quelque algorithme que ce soit dans les logiciels utilisés par la DGA, ceci nécessitant le déplacement du programmeur sur les sites même de la DGA et nuisant alors au bon déroulement de la thèse. Ceci étant, l'ensemble de ces travaux ont reçu un avis favorable par la commission de sélection des bourses DGA.

Plan du Mémoire

Dans un premier chapitre, nous présentons quelques outils fondamentaux de la géométrie discrète et de la géométrie algorithmique. À cette occasion, nous introduisons les espaces discrets comme des grilles régulières dans lesquelles les objets sont toujours exprimés grâce à des nombres entiers. Nous présentons également un groupe de transformations particulières dites transformations unimodulaires ou transvections permettant de transformer les objets discrets dans des espaces numériques. Par la suite, nous revenons en détail sur la théorie des nombres, une branche des mathématiques qui étudie les propriétés combinatoires des nombres entiers. Nous étudions dans cette partie l'identité de Bezout et les fractions continues et nous cherchons à corréler leur interprétation numérique et géométrique. Dans cette même section, nous présentons quelques unes des principales méthodes de construction d'enveloppes convexes dans le plan. En effet, la construction d'enveloppes convexes est un outil fondamental en géométrie algorithmique qui mérite une attention particulière dans ce mémoire. À la fin du premier chapitre, nous dédions une section au

théorème de Pick, fondamental pour le dénombrement en géométrie discrète, et nous présentons brièvement un problème bien connu en programmation linéaire en nombres entiers appelé *problème du sac à dos* dont l'étude est liée au calcul d'enveloppes convexes sur des grilles discrètes. Ces différents outils sont utilisés dans les résultats décrits dans le chapitre suivant.

Dans le chapitre 2, nous présentons un des principaux résultats obtenus durant cette thèse. Nous décrivons l'algorithme proposé pour déterminer le nombre rationnel à dénominateur borné qui approxime au mieux un nombre réel a donné. Comme expliqué en amont, l'équivalent géométrique de ce problème consiste à déterminer le point à coordonnées entières situé dans un domaine vertical D de la forme $\{(x, y) \in \mathbb{R}^2 | b \leq x \leq b'\}$ tel que la droite passant par l'origine et par ce point approxime au mieux la droite L de pente a passant par l'origine. Notre algorithme consiste à déterminer les sommets des enveloppes convexes des points de la grille situés au-dessus ou en dessous de la droite L et appartenant au domaine vertical D. Après une description précise du problème, nous présentons deux méthodes efficaces existantes pour calculer ces enveloppes et nous décrivons notre nouvel algorithme. Enfin, nous mettons en évidence deux applications de notre méthode. L'une d'entre elles est la reconstruction d'un polygone convexe à sommets portés par la grille après une coupe par une droite quelconque.

Nous proposons alors dans le chapitre 3 différentes manières de définir un morceau de plan discret, et en particulier un morceau de plan discret naïf. Nous détaillons alors un panel de différentes méthodes existantes pour la reconnaissance de plans discrets. Cet état de l'art recense les algorithmes suivant trois catégories : ceux utilisant la géométrie algorithmique et en particulier des calculs d'enveloppes convexes, ceux mettant en oeuvre la programmation linéaire et enfin ceux n'entrant dans aucune des deux premières catégories comme l'algorithme utilisant le critère de régularité. Cette état de l'art permet de mettre le doigt sur les déficiences des méthodes existantes avant d'exposer notre propre algorithme dans le chapitre suivant.

Nous décrivons donc en détail dans le chapitre 4 notre nouvelle méthode de reconnaissance de plans discrets naïfs. Cette méthode revient à résoudre un problème de réalisabilité sur une fonction convexe bi-dimensionnelle. Nous décrivons et justifions chaque étape de notre algorithme puis nous étudions sa complexité en temps. Nous décrivons de plus une version incrémentale de notre méthode. Nous proposons enfin des résultats expérimentaux mettant en évidence l'efficacité en pratique de notre algorithme dans le cas incrémental et non-incrémental grâce à des améliorations pratiques.

Le chapitre 5 est un peu détaché du reste du mémoire puisqu'il présente les résultats obtenus lors de notre collaboration avec F. Feschet. Nous définissons dans ce chapitre l'épaisseur d'un objet discret par rapport au réseau classique $\mathbb{Z}^d$ et nous décrivons l'algorithme proposé par F. Feschet dans [Fes06] pour calculer une telle épaisseur dans le plan. Nous détaillons alors notre nouvelle méthode, d'abord dans le cas planaire où la complexité linéaire optimale est atteinte, puis en toute dimension avec une approche gloutonne efficace.

Chapitre 1

Préliminaires

1.1 Espaces Discrets et Transformations Unimodulaires

1.1.1 Introduction aux Espaces Discrets

En géométrie discrète, tous les calculs se font avec des entiers. Les objets géométriques que nous manipulons comme les points, les droites ou encore les plans doivent donc toujours pouvoir être exprimés grâce à des entiers. Ceci implique qu'un point en géométrie discrète a toujours des coordonnées entières et appartient donc à $\mathbb{Z}^d$. Nous parlons généralement de *point entier* ou de *point discret* pour désigner un point à coordonnées entières. En dimension 2 et 3, les points entiers sont généralement assimilés respectivement à des *pixels* et à des *voxels*. Le pixel correspond à l'unité de surface utilisée pour mesurer les images numériques et son nom vient de l'anglais *picture element*. Il correspond généralement à un rectangle mais peut être assimilé à un carré dans le cadre de notre étude. Dans ce cas, lorsqu'on considère les points d'une image discrète cela revient à considérer les centres des pixels qui composent l'image. Le voxel est l'analogue 3D du pixel.

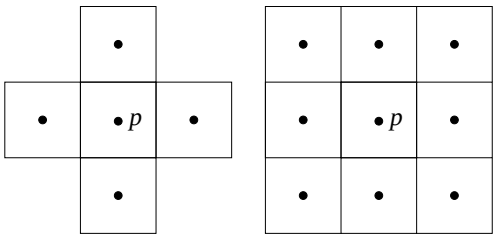
En dimension 2, nous définissons le 4-voisinage et le 8-voisinage des points entiers, et donc des pixels. Le 4-voisinage d'un point p de $\mathbb{Z}^2$ est noté $V_4(p)$ et correspond aux quatre points de $\mathbb{Z}^2$ obtenus à partir de p grâce à un déplacement horizontal ou vertical. Le 8-voisinage d'un point p de $\mathbb{Z}^2$ est noté $V_8(p)$ et correspond aux points de $V_4(p)$ auxquels nous ajoutons les quatre points de $\mathbb{Z}^2$ obtenus à partir de p grâce à un déplacement en diagonale. Nous rappelons que la *norme 1* d'un vecteur $u = (u_1, \cdots, u_d)$ notée $||u||_1$ est égale à $|u_1|+\cdots+|u_d|$ et que sa *norme à l'infini* notée $||u||_\infty$ est égale à $\max(|u_1|, \cdots, |u_d|)$. La figure 1.1 illustre ces voisinages 2D. Plus formellement, les voisinages en 2D sont définis de la manière suivante :

$$\text{Pour tout } p \in \mathbb{Z}^2, V_4(p) = \{q \in \mathbb{Z}^2 : q \neq p, ||pq||_1 \leq 1\}$$

$$\text{Pour tout } p \in \mathbb{Z}^2, V_8(p) = \{q \in \mathbb{Z}^2 : q \neq p, ||pq||_\infty \leq 1\}$$

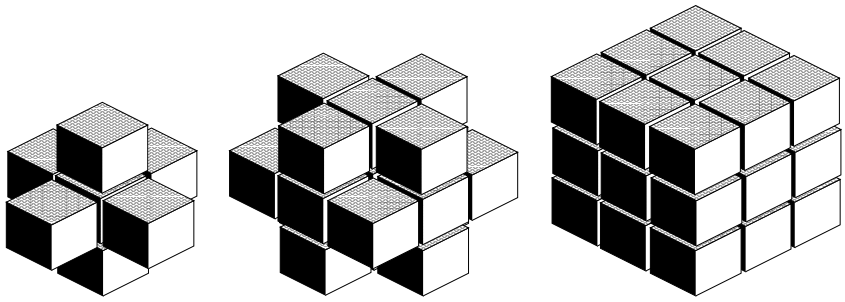
De façon analogue, les points entiers en 3D admettent un 6, 18 et 26-voisinage (voir figure 1.2) définis de la manière suivante :

$$\text{Pour tout } p \in \mathbb{Z}^3, V_6(p) = \{q \in \mathbb{Z}^3 : q \neq p, ||pq||_1 \leq 1\}$$



(a) $V_4(p)$ (b) $V_8(p)$

FIG. 1.1 – Voisinage des points en 2D



(a) $V_6(p)$ (b) $V_{18}(p)$ (c) $V_{26}(p)$

FIG. 1.2 – Voisinage des points en 3D

$$\text{Pour tout } p \in \mathbb{Z}^3, V_{18}(p) = \{q \in \mathbb{Z}^3 : q \neq p, ||pq||_\infty \leq 1 \text{ et } ||pq||_1 \leq 2\}$$

$$\text{Pour tout } p \in \mathbb{Z}^3, V_{26}(p) = \{q \in \mathbb{Z}^3 : q \neq p, ||pq||_\infty \leq 1\}$$

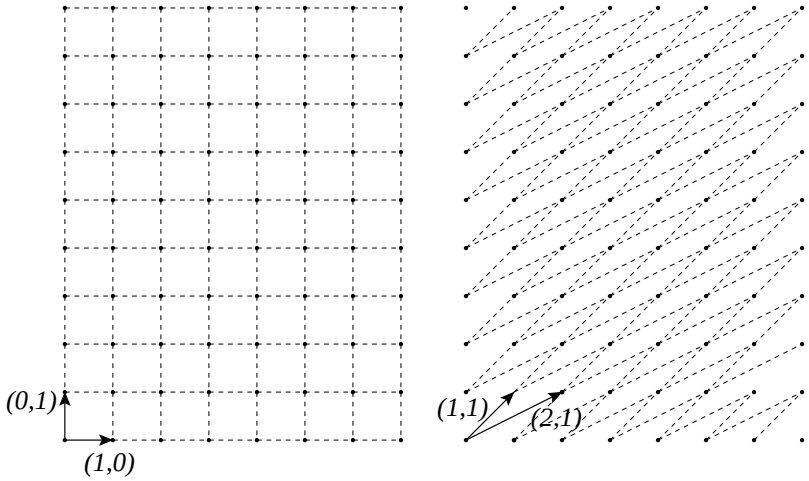
Deux points p et q de $\mathbb{Z}^d$ sont *k-voisins* si $q \in V_k(p)$. Ceci implique alors que $p \in V_k(q)$.

L'ensemble des points discrets 2D et 3D sont généralement disposés sur une grille régulière. Cependant, le repère de cette grille n'est pas obligatoirement orthonormée et les vecteurs de la base du repère ne définissent donc pas forcément la matrice identité. En effet, les d vecteurs $b_1, b_2, \cdots, b_d$ forment une base de $\mathbb{Z}^d$ si et seulement si le déterminant de la matrice dont les colonnes correspondent aux vecteurs b_i, $1 \leq i \leq d$, vaut 1 (voir la figure 1.3). Tout point de la grille discrète peut être exprimé comme combinaison linéaire à coefficients entiers des vecteurs de la base. En 2D, les vecteurs de la base définissent toujours un parallélogramme d'aire 1.

1.1.2 Transvections

Nous nous intéressons dans cette section aux transformations particulières dites *transvections* (en anglais *shear transforms*). Nous rappelons qu'une matrice est dite *unimodulaire* si son déterminant est égal à 1. De plus, nous rappelons qu'une *transformation affine* est une transformation entre deux espaces affines qui préserve la notion de parallélisme et la structure affine, c'est-à-dire qui envoie des droites sur des droites, des plans sur des plans et ainsi de suite.

Définition 1 *Une* transvection *est une transformation associée à une matrice carrée T_{ij} de la forme $I_n + \alpha_{ij}E_{ij}$ où I_n correspond à la matrice identité et où E_{ij} correspond à la matrice nulle à l'exception du terme situé à la i-ième ligne et à la j-ième colonne qui*



(a) Base $((1,0),(0,1))$ (b) Base $((2,1),(1,1))$

FIG. 1.3 – Exemples de repères pour les grilles

vaut 1. *La valeur* α_{ij} *est appelée le* coefficient de la transvection. *La matrice associée est toujours de déterminant* 1 *et s'écrit :*

$$T_{ij} = \begin{pmatrix} 1 & 0 & \cdots & & & 0 \\ 0 & 1 & \cdots & & & \vdots \\ \vdots & & \ddots & \alpha_{ij} & & \\ & & & \ddots & & \\ & & & & \ddots & 0 \\ 0 & \cdots & & \cdots & 0 & 1 \end{pmatrix}$$

Les transvections sont des transformations affines unimodulaires qui transforment les images en les "étirant" dans une direction et en laissant invariant un hyperplan. Les matrices de transvection sont donc inversibles et leur inverse correspond à la matrice d'une transvection dans le sens opposé. Les transvections sont des applications bijectives. Dans le cas bimensionnel par exemple, nous avons des transvections horizontales (voir figure 1.4) d'équations et de matrice correspondantes :

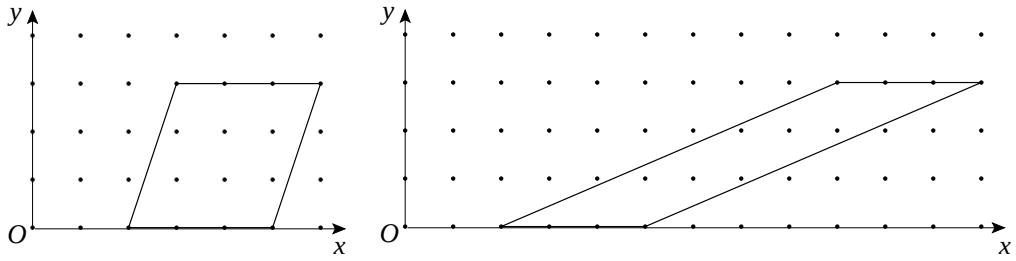
$$\begin{cases} x' = x + \alpha y \\ y' = y \end{cases} \quad \Leftrightarrow \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

et des transvections verticales (voir figure 1.5) d'équations et de matrice correspondantes :

$$\begin{cases} x' = x \\ y' = y + \alpha x \end{cases} \quad \Leftrightarrow \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

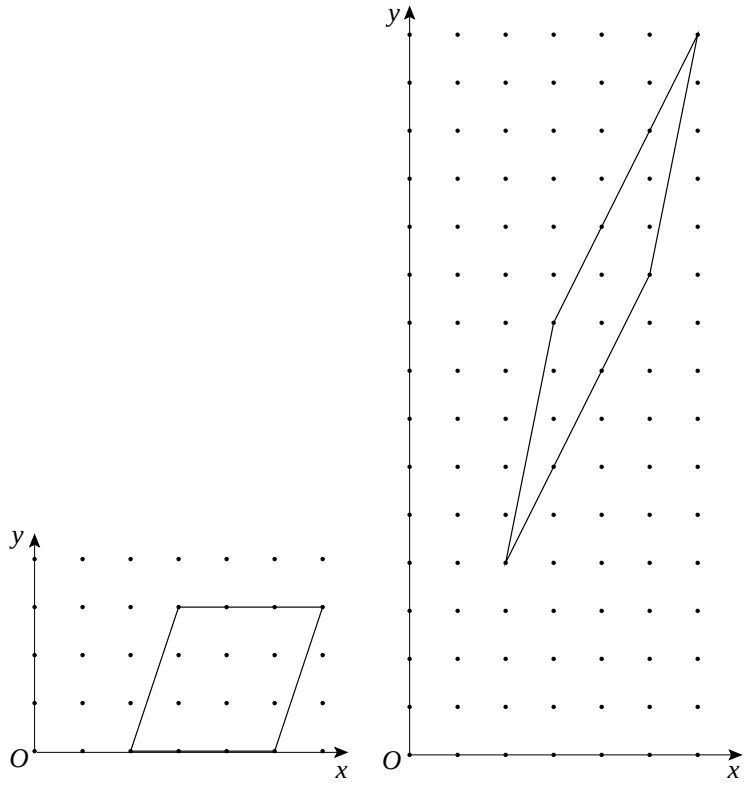
Les transvections admettent certaines propriétés intéressantes, parmi lesquelles :

Proposition 1 *Les matrices de transvection d'un espace vectoriel* E *de dimension* n *génèrent le* groupe spécial linéaire *d'ordre* n *noté* $SL_n(E)$.



(a) Image initiale (b) Image étirée

FIG. 1.4 – Exemple de transvection horizontale de coefficient 2



(a) Image initiale (b) Image étirée

FIG. 1.5 – Exemple de transvection verticale de coefficient 2

Nous rappelons que le groupe spécial linéaire d'ordre n d'un espace vectoriel E correspond à l'ensemble des matrices inversibles à coefficients dans E de déterminant 1. Cette proposition signifie en d'autres termes que toute matrice unimodulaire correspond au produit d'un nombre fini de matrices de transvection.

Propriété 1 *Les transvections conservent les orientations et les aires comme les volumes.*

Nous démontrons cette propriété dans le cas bi-dimensionnel pour plus de clarté. Cependant, une démonstration similaire est également valable en dimensions supérieures. Soient $u = (u_1, u_2)$ et $v = (v_1, v_2)$ deux vecteurs de $\mathbb{Z}^2$, nous rappelons que l'expression $u \wedge v$ avec est égale à $u_1v_2 - u_2v_1$. Cette expression est appelée produit vectoriel des vecteurs u et v et cette valeur correspond à l'aire signée du parallélogramme de côté u et v. Nous parlons d'aire signée car cette valeur est positive ou négative selon le signe de l'angle formé entre les vecteurs u et v. Soit T une transvection de $\mathbb{Z}^2$, supposons par exemple qu'il s'agisse d'une transvection horizontale de coefficient α. Les images de u et de v par la transvection T sont de la forme $T(u) = (u_1 + \alpha u_2, u_2)$ et $T(v) = (v_1 + \alpha v_2, v_2)$. Nous vérifions donc facilement que $T(u) \wedge T(v)$ est égale à $u \wedge v$. En effet, nous avons bien :

$$\begin{aligned} T(u) \wedge T(v) &= (u_1 + \alpha u_2)v_2 - u_2(v_1 + \alpha v_2) \\ &= u_1v_2 + \alpha u_2v_2 - u_2v_1 - \alpha u_2v_2 \\ &= u \wedge v \end{aligned}$$

Le fait que le produit vectoriel soit conservé signifie d'une part que l'aire des parallélogrammes, et donc des triangles, est conservée par transvection. Tout polygone pouvant être décomposé en un ensemble de triangles, l'aire des polygones est conservée par transvection horizontale. Dans l'exemple illustré par la figure 1.4, nous remarquons que le parallélogramme est bien d'aire 9 avant et après la transvection horizontale de coefficient 2. Cette démonstration est également valable pour les transvections verticales. Par conséquent, l'application d'un nombre fini de transvections verticales et horizontales conserve bien les aires en 2D.

Nous rappelons par ailleurs que l'expression $u \wedge v$ peut également être écrite sous la forme $||u|| * ||v|| * \sin(u, v)$ où $\sin(u, v)$ correspond au sinus de l'angle entre le vecteur u et le vecteur v, décrit de u vers v. Nous rappelons qu'en géométrie les angles sont signés suivant le sens trigonométrique et que les normes sont toujours positives. Le sinus d'un angle positif est positif et de même, le sinus d'un angle négatif est négatif. Le fait que le produit vectoriel soit conservé par transvection signifie donc d'autre part que le signe du produit vectoriel est conservé. Par conséquent, le signe des angles reste le même avant et après transvection, c'est ce qu'on appelle la conservation des orientations par transvection. Par extension, nous remarquons facilement que le positionnement des points les uns par rapport aux autres est fortement lié avec la notion d'orientation des angles. En effet, considérons deux points P et P' situés de part et d'autre d'une droite AB. Dans ce cas, le signe de l'angle entre les vecteurs AB et AP est différent du signe de l'angle entre les vecteurs AB et AP'. En outre, si un point P'' est porté par la droite AB alors le produit vectoriel des vecteurs AB et AP'' est nul. Après une transvection, le produit vectoriel étant inchangé, le positionnement des points par rapport à la droite est conservé de même que l'alignement des points.

Dans la section 1.3.1, nous définissons l'enveloppe convexe d'un ensemble de points. Cette définition apparaît plus tard dans le mémoire, mais nous en donnons une première

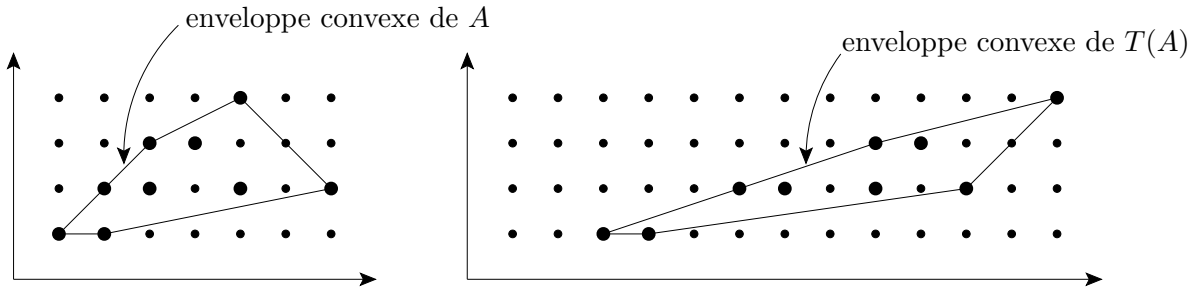


FIG. 1.6 – Conservation de la convexité des enveloppes par transvection

définition en $2D$ dès à présent. Soit $A = (a_1, a_2, \cdots, a_n)$ un ensemble de n points dans le plan, l'enveloppe convexe de l'ensemble A correspond à un polygone C de sommets $(c_1, c_2, \cdots, c_m)$ défini comme suit. L'ensemble des sommets de C est inclus dans A, il est décrit dans le sens inverse des aiguilles d'une montre et pour tout i tel que $1 \leq i \leq m-1$, tous les points de l'ensemble A se situent du même côté de l'arête $c_i c_{i+1}$. Grâce à la propriété de conservation des orientations par transvection et par définition des enveloppes convexes, nous en déduisons qu'après transformation tous les points de $T(A)$ seront situés du même côté de chaque arête $T(c_i c_{i+1})$. Ceci justifie que chaque arête transformée correspond à une arête de l'enveloppe convexe de $T(A)$. La convexité des enveloppes est donc conservée par transvection : soit C l'enveloppe convexe de l'ensemble A et soit T une transvection, alors $T(C)$ correspond à l'enveloppe convexe de l'ensemble $T(A)$. Nous pouvons donc appliquer une ou plusieurs transvections sur un ensemble de points de Z^2 par exemple, calculer son enveloppe convexe puis appliquer la transformation inverse afin de retrouver l'enveloppe convexe de l'ensemble initial. Dans l'exemple de la figure 1.6, A correspond à l'ensemble des points en gras. Nous calculons son enveloppe convexe C et nos appliquons une transvection horizontale de coefficient 2. L'enveloppe convexe de l'ensemble A est bien obtenue en appliquant la transvection inverse T^{-1} sur l'enveloppe convexe de $T(A)$.

Dans la section 1.1.1, nous indiquions que tous les points d'une grille régulière devaient pouvoir être exprimés comme combinaison linéaire entière des d vecteurs définissant la base du repère. Nous nous apercevons donc que la base orthonormée en dimension d peut toujours être transformée en une autre base en appliquant une succession de transvections, la base orthonormée pouvant alors être retrouvée en appliquant les transvections inverses.

1.2 Outils Généraux de la Théorie des Nombres

1.2.1 Identité de Bezout

Du Point de Vue Numérique

En théorie des nombres, l'*identité de Bezout* correspond à une équation linéaire diophantienne particulière. Soient a et b deux valeurs entières non nulles, alors il existe deux entiers x et y tels que $ax + by = pgcd(a, b)$, où $pgcd(a, b)$ correspond au plus grand diviseur commun de a et b. Plus généralement, l'équation linéaire diophantienne $ax + by = c$ admet une infinité de solutions entières si et seulement si le $pgcd$ de a et b divise c ; sinon, il n'y en a aucune. L'ensemble des solutions est donné par :

$$\left\{\left(x_0 + \frac{kb}{\gcd(a,b)},\ y_0 - \frac{ka}{\gcd(a,b)}\right) \mid k \in \mathbb{Z}\right\} \tag{1.1}$$

où (x_0, y_0) correspond à une solution particulière de l'équation. Pour de plus amples détails sur l'identité de Bezout, vous pouvez consulter [Mor02].

Notons qu'une solution particulière d'une équation diophantienne peut être obtenue en appliquant l'algorithme d'Euclide étendu avec a et b (voir [Knu68]). L'algorithme d'Euclide dans sa version simple est utilisé pour calculer le *pgcd* de deux nombres entiers (voir [Sha94]). L'algorithme 1 résume cette méthode. Nous avons choisi une présentation itérative de l'algorithme bien que celui-ci puisse également être formalisé de manière récursive. Nous supposons que les nombres a et b passés en paramètres sont des entiers positifs. La méthode consiste à générer une suite d'entiers $R = (r_0, r_1, \cdots, r_n)$ dont les termes r_0 et r_1 sont initialisés respectivement à a et b lorsque a est supérieure à b ou à b et a lorsque b est strictement supérieur à a. Chaque terme r_k correspond au reste de la division euclidienne de r_{k-2} par r_{k-1}. Le dernier terme non nul de la suite R est égal au *pgcd* des nombres passés en paramètres. Nous pouvons facilement montrer que la complexité dans le pire cas de l'algorithme d'Euclide est atteinte pour deux termes consécutifs de la *suite de Fibonacci*. Nous rappelons que la suite de Fibonacci $(F_i)_{i\geq 0}$ est une suite d'entiers initialisée par $F_0 = 0$ et $F_1 = 1$ et définie par la relation de récurrence linéaire $F_{i+1} = F_i + F_{i-1}$ pour $i \geq 1$. Cette suite est positive et strictement croissante. Nous remarquons que dans la formule de récurrence F_{i-1} correspond au reste de la division euclidienne de F_{i+1} par F_i. Par conséquent, en exécutant l'algorithme d'Euclide avec deux termes consécutifs F_k et F_{k+1} de la suite de Fibonacci en entrée, la suite de restes générée correspond exactement à tous les termes de la suite de Fibonacci d'indices strictement inférieurs à k. Le *théorème de Lamé* nous permet de calculer la complexité de l'algorithme d'Euclide :

Théorème 1 *Pour tout entier $k \geq 1$ et tout entier a et b tels que $a \geq b \geq 0$, si $b < F_{k+1}$, alors l'algorithme d'Euclide sur a et b exécute moins de k itérations. De plus, si $a = F_{k+2}$ et $b = F_{k+1}$ alors l'algorithme effectue exactement k itérations.*

Reste alors à déterminer une borne supérieure pour k. Soit ϕ le *nombre d'or* de valeur $(1+\sqrt{5})/2$, la *formule de Binet* nous indique que le terme F_k vérifie :

$$F_k = \frac{1}{\sqrt{5}}(\phi^k - (-\phi)^{-k})$$

Il s'en suit que :

$$\begin{aligned} F_{k+1} &= \frac{1}{\sqrt{5}}(\phi^{k+1} + (-1)^k(\frac{1}{\phi})^{k+1}) \\ &= \frac{1}{\sqrt{5}}\phi^{k+1}(1 + (-1)^k(\frac{1}{\phi})^{2(k+1)}) \end{aligned}$$

et donc que :

$$F_{k+1} \geq \frac{1}{\sqrt{5}}\phi^{k+1}(1 - (\frac{1}{\phi})^{2(k+1)})$$

Nous en déduisons que :

$$\ln(F_{k+1}) \geq -\ln(\sqrt{5}) + (k+1)\ln(\phi) + \ln(1 - (\frac{1}{\phi})^{2(k+1)})$$

ce qui nous donne :

$$k \leq \frac{\ln(F_{k+1})}{\ln(\phi)} + \frac{\ln(\sqrt{5})}{\ln(\phi)} - 1 - \frac{\ln(1 - (\frac{1}{\phi})^{2(k+1)})}{\ln(\phi)}$$

Puisque $k \geq 1$, nous pouvons réécrire la formule comme :

$$k \leq \frac{\ln(F_{k+1})}{\ln(\phi)} + \frac{\ln(\sqrt{5})}{\ln(\phi)} - 1 - \frac{\ln(1 - (\frac{1}{\phi})^{4})}{\ln(\phi)}$$

Nous obtenons donc :

$$k \leq 2.0781 \ln(F_{k+1}) + 1$$

Dans ce cas, le nombre d'itérations de l'algorithme d'Euclide est donc borné par le logarithme du terme F_{k+1}. De façon générale, l'algorithme d'Euclide de paramètres d'entrée a et b s'exécute donc en $O(\max(\log(a), \log(b)))$.

Algorithme 1 : Algorithme d'Euclide

```
PGCD(a : entier, b : entier)
Valeur retournée : entier
1  SI a < b
2      Inverser(a,b)
3  r_0 ← a
4  r_1 ← b
5  k ← 1
6  TANT QUE r_k ≠ 0
7      r_{k+1} ← r_{k-1} % r_k
8      k ← k + 1
9  Retourner r_{k-1}
```

La version étendue de l'algorithme d'Euclide fonctionne de façon similaire et a la même complexité en temps mais elle permet en plus de calculer une solution entière (x, y) de l'équation diophantienne $ax+by = pgcd(a, b)$. Pour cela, en plus de calculer la suite entière R des restes successifs, nous calculons la suite des quotients $Q = (q_2, \cdots, q_{n-1})$ qui servira à déterminer les suites $X = (x_0, x_1, \cdots, x_{n-1})$ et $Y = (y_0, y_1, \cdots, y_{n-1})$. Chaque terme q_k correspond au quotient de la division euclidienne de r_{k-2} par r_{k-1} et les termes des suites X et Y sont définis comme suit :

$$x_0 = 1, x_1 = 0$$

$$y_0 = 0, y_1 = 1$$

$$x_k = x_{k-2} - q_{n-2} * x_{n-1}$$

$$y_k = y_{k-2} - q_{n-2} * y_{n-1}$$

L'algorithme se termine lorsqu'un reste nul est calculé et les termes x_{n-1} et y_{n-1} vérifient $ax_{n-1} - by_{n-1} = pgcd(a, b)$. Cette méthode est présentée dans l'Algorithme 2

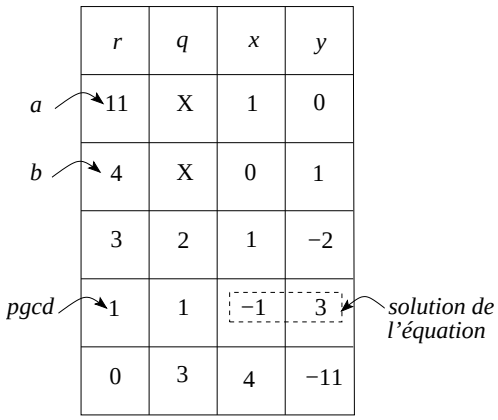


FIG. 1.7 – Exemple de déroulement de l'algorithme d'Euclide étendu

et la figure 1.7 en illustre un exemple de déroulement avec $a = 11$ et $b = 4$. Dans cet exemple, le *pgcd* de 11 et 4 correspond au dernier reste non-nul 1 et le couple de valeurs entières $(-1, 3)$ vérifie bien $11 * (-1) + 4 * 3 = 1$.

Algorithme 2 : Algorithme d'Euclide étendu

PGCD(a : entier, b : entier)
Valeur retournée : entier
1 SI $a < b$
2 Inverser(a,b)
3 $r_0 \leftarrow a, r_1 \leftarrow b$
4 $x_0 \leftarrow 1, x_1 \leftarrow 0$
5 $y_0 \leftarrow 0, y_1 \leftarrow 1$
6 $k \leftarrow 1$
7 TANT QUE $r_k \neq 0$
8 $q_{k+1} \leftarrow r_{k-1}/r_k$
9 $r_{k+1} \leftarrow r_{k-1}\%r_k$
10 $x_{k+1} \leftarrow x_{k-1} - q_k * x_k$
11 $y_{k+1} \leftarrow y_{k-1} - q_k * y_k$
12 $k \leftarrow k + 1$
13 Retourner r_{k-1}

Du Point de Vue Géométrique

L'algorithme d'Euclide peut également être visualisé géométriquement. Soient a et b les nombres entiers dont nous souhaitons calculer le *pgcd*, le principe est de construire un rectangle de longueur a et de largeur b. Par la suite, nous partitionnons ce rectangle en un minimum de carrés. La longueur du côté du plus petit carré correspond au *pgcd* de a et b. Dans l'exemple illustré par la figure 1.8, nous retrouvons bien que le *pgcd* de 11 et 4 vaut 1. En effet, dans le rectangle de dimensions 11×4, nous commencons par construire deux carrés de côté 4. Ensuite, dans le rectangle restant de dimensions 4×3, nous construisons un carré de côté 3. Enfin, dans le rectangle de dimensions 3×1, nous insérons trois carrés de côté 1. Cette dernière longueur correspond bien au *pgcd* à calculer.

Un vecteur à coordonnées entières dans le plan euclidien $u = (u_1, u_2)$ est *irréductible* si le *pgcd* de ses coordonnées u_1 et u_2 est égal à 1. Soient $u = (u_1, u_2)$ et $v = (v_1, v_2)$

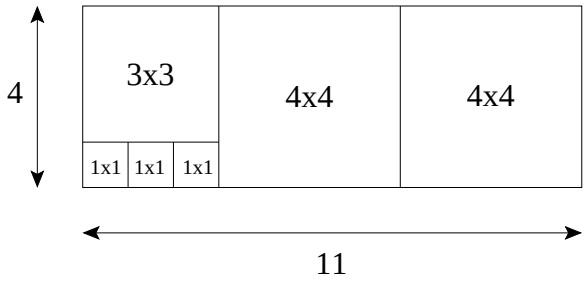


FIG. 1.8 – Algorithme d'Euclide géométrique

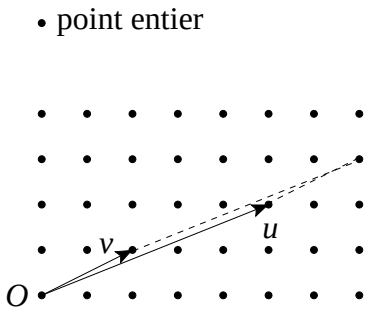


FIG. 1.9 – Vecteurs de Bezout $u = (5, 2)$ et $v = (2, 1)$

deux vecteurs entiers, la valeur $u \wedge v$ est égale à $u_1v_2 - u_2v_1$. Cette valeur correspond au produit vectoriel de u et v et elle est égale à l'aire signée du parallélogramme généré par les vecteurs u et v.

Définition 2 *Soit u un vecteur entier irréductible. Nous appellons* vecteur de Bezout *de u un vecteur entier v tel que $u \wedge v = 1$. Ceci signifie que le paralélogramme $(0, u, u + v, v)$ ne contient aucun point entier à part ses sommets.*

Par définition, un vecteur de Bezout est donc forcément un vecteur irréductible. Étant donné un vecteur entier irréductible $u = (u_1, u_2)$, déterminer l'ensemble de ses vecteurs de Bezout de la forme (v_1, v_2) revient à résoudre l'équation diophatienne $u_1v_2 - u_2v_1 = 1$. Cette équation admet toujours une infinité de solutions entières. Soit v une solution particulière de cette équation, nous en déduisons que l'ensemble des vecteurs de Bezout de u est de la forme $\{v+ku|k \in \mathbb{Z}\}$. En effet, nous vérifions bien que : $u\wedge(v+ku) = u\wedge v = 1$. La figure 1.9 illustre un exemple de vecteurs de Bezout avec $u = (5, 2)$ et $v = (2, 1)$ un vecteur de Bezout de u. Nous vérifions bien que $u \wedge v = 1$ et que le parallélogramme $(O, u, u + v, v)$ ne contient aucun point entier.

Remarque 1 *Si le vecteur v est un vecteur de Bezout de u alors il n'y a aucun point entier strictement inclus dans le parallélogramme de la forme $(O, v, v + ku, ku)$, où k correspond à une valeur positive entière.*

Remarque 2 *Deux vecteurs de Bezout u et v forment une base. Ceci signifie que tous les points entiers de la grille peuvent être exprimés comme une combinaison linéaire entière des vecteurs u et v. Nous remarquons que, par transformation unimodulaire entière, deux vecteurs de Bezout peuvent toujours être ramenés à une base orthonormée $(1, 0)$ et $(0, 1)$.*

1.2.2 Fractions Continues

Dès le Moyen Âge, des mathématiciens indiens utilisaient les fractions continues. Elles apparaissent ensuite en Europe au XVII^{e} siècle et sont encore de nos jours largement étudiées. Nous introduisons dans cette section la définition des fractions continues ainsi que différentes propriétés interprétées du point de vue numérique et du point de vue géométrique.

Interprétation Numérique

Une introduction aux fractions continues plus détaillée est disponible dans [Chr89] ou [Dav92].

Définition 3 *La* fraction continue simple *d'un nombre réel x correspond à :*

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots}}}}$$

où a_0 représente une valeur entière et où chaque a_i représente une valeur positive entière pour $i \geq 1$. Habituellement, nous utilisons la notation $x = [a_0, a_1, a_2, \cdots]$ et le terme a_i est appelé le coefficient d'indice *i de la fraction continue.*

Le calcul de la décomposition en fraction continue d'un rationnel a/b se fait en exécutant l'algorithme d'Euclide étendu introduit dans la section 1.2.1 avec a et b en paramètres [FV98]. En effet, les quotients de la série $Q = (q_2, \cdots, q_{n-1})$ générée lors du déroulement de l'algorithme correspondent aux coefficients de la fraction continue.

Définition 4 *Les* convergents principaux *d'un nombre réel x correspondent à leurs approximations rationnelles p_k/q_k. Ils sont obtenus en tronquant la décomposition en fractions continues de x après le coefficient d'indice k. Le numérateur et le dénominateur des convergents principaux sont calculés de la manière suivante :*

$$\begin{cases} p_0 = a_0 & p_1 = a_0 a_1 + 1 & p_{k+2} = p_k + a_{k+2} p_{k+1} \\ q_0 = 1 & q_1 = a_1 & q_{k+2} = q_k + a_{k+2} q_{k+1} \end{cases}$$

Chaque convergent d'indice pair est plus petit que le réel x et chaque convergent d'indice impair est plus grand que le réel x. La valeur de chaque convergent est plus proche de x que le convergent précédent.

Définition 5 *Les* convergents intermédiaires *entre deux convergents principaux p_k/q_k et p_{k+2}/q_{k+2} sont définis de la manière suivante :*

$$\frac{p_k + i p_{k+1}}{q_k + i q_{k+1}}, \quad i = 1 \cdots a_k - 1$$

Notons S_P et S_I les suites définies respectivement par tous les convergents principaux d'indice pair et d'indice impair, intercalés avec leurs convergents intermédiaires. Nous ajoutons respectivement les fractions $0/1$ et $1/0$ au début de chacunes des suites S_P et S_I si elles ne commencent pas déjà par ces fractions. Lorsque x est un nombre rationnel, alors une de ces deux suites ne se termine pas par x. Dans ce cas, nous ajoutons le nombre rationnel x à la fin de cette suite. Nous utilisons les notations suivantes : $[S_{P_1}, S_{P_2}, \cdots]$

correspond à la suite S_P et $[S_{I_1}, S_{I_2}, \cdots]$ correspond à la suite S_I. Nous remarquons que la suite S_P (resp. S_I) est strictement croissante (resp. strictement décroissante) et que la suite des dénominateurs de S_P et de S_I est croissante.

Une autre façon de visualiser le développement en fractions continues d'un nombre rationnel consiste à utiliser l'arbre de Stern-Brocot (voir [GKP94]). L'arbre de Stern-Brocot a été découvert il y a plus de 150 ans simultanément par un mathématicien allemand nommé Moritz Stern et par un horloger français répondant au nom d'Achille Brocot. L'arbre de Stern-Brocot est une représentation de toute les fractions irréductibles positives. Pour le construire, nous pouvons imaginer que nous écrivons respectivement les fractions $0/1$ et $1/0$ à l'extrême gauche et à l'extrême droite. Ensuite, nous intercalons entre deux fractions leur *fraction médiante* de façon à construire un arbre de racine $1/1$. La fraction médiante de deux fraction a/b et c/d correspond à la fraction $(a+c)/(b+d)$. Pour retrouver une fraction dans l'arbre de Stern-Brocot, nous utilisons son développement en fraction continue qui détermine le chemin à parcourir depuis la racine de l'arbre pour atteindre la fraction. Soit $[a_0, a_1, \cdots, a_m]$ le développement en fraction continue d'un rationnel x. Les coefficients d'indice pair indiquent un nombre de déplacements vers la droite dans l'arbre de Stern-Brocot. Les coefficients d'indice impair indiquent un nombre de déplacements vers la gauche dans ce même arbre. Une exception pour le dernier coefficient a_m auquel il faut retrancher 1 pour obtenir un nombre de déplacements. Considérons par exemple le rationnel $x = 7/4$ de développement en fraction continue $[1, 1, 3]$. Pour le retrouver dans l'arbre de Stern-Brocot à partir de la racine, il faut donc effectuer un déplacement vers la droite, puis un déplacement vers la gauche et enfin deux déplacements vers la droite (voir la figure 1.10). Ces déplacements dans l'arbre de Stern-Brocot peuvent également être vus de façon matricielle. Soit p/q une fraction dans l'arbre de Stern-Brocot. Cette fraction peut être représentée comme le vecteur $(p, q)^T$. Nous obtenons la fraction située à gauche de p/q dans l'arbre de Stern-Brocot en multipliant le vecteur $(p, q)^T$ par la matrice de transvection verticale de coefficient 1. De la même manière, nous obtenons la fraction située à droite de p/q dans l'arbre de Stern-Brocot en multipliant le vecteur $(p, q)^T$ par la matrice de transvection horizontale de coefficient 1. Ainsi, soit $[a_0, a_1, \cdots, a_m]$ le développement en fraction continue d'un rationnel $x = r/s$, nous vérifions que :

$$\begin{pmatrix} 1 & a_0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ a_1 & 1 \end{pmatrix} \begin{pmatrix} 1 & a_2 \\ 0 & 1 \end{pmatrix} \cdots \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}$$

Nous énonçons une proposition qui sera déterminante par la suite pour résoudre le problème d'approximation du Chapitre 2 (voir [Chr89] pour la preuve) :

Proposition 2 *Soit r le nombre rationnel plus petit (resp. plus grand) qu'un nombre réel a, qui approxime au mieux a et tel que son dénominateur n'excède pas un entier donné d. Le nombre rationnel r est le terme de la suite S_P (resp. S_I) de a de plus grand dénominateur n'excédant pas d.*

Interprétation Géométrique

Dans le plan euclidien, nous pouvons établir une correspondance entre le nombre rationnel p/q et le point à coordonnées entières (q, p). En effet, la droite d'équation $y = (p/q)x$ passe par le point de coordonnées (q, p). Ainsi, nous pouvons interpréter géométriquement la fraction continue associée à un nombre réel a en faisant correspondre les nombres rationnels avec des points entiers. De cette façon, les convergents principaux

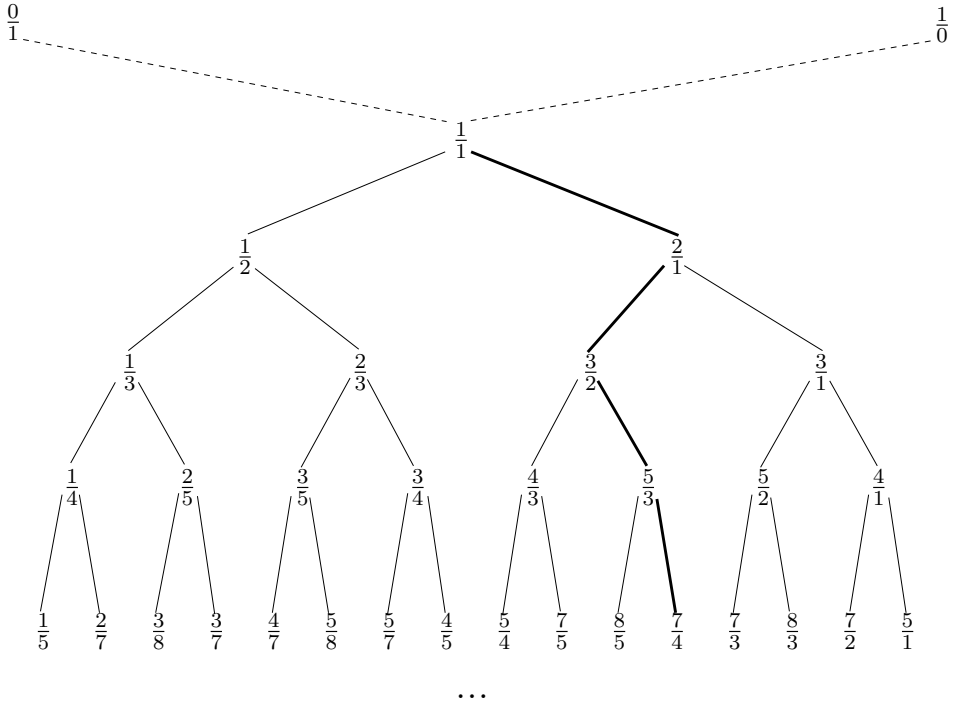


FIG. 1.10 – Arbre de Stern-Brocot

d'ordre pair et leur convergents intermédiaires associés étant plus petits que le réel de référence a, ils correspondent à des points entiers situés en dessous de la droite L d'équation $y = ax$. De la même manière, les convergents principaux d'ordre impair et leur convergents intermédiaires associés étant plus grands que le réel de référence a, ils correspondent à des points entiers situés au-dessus de la droite L. Nous établissons alors une correspondance entre la suite de rationnels $S_P = [S_{P_1}, S_{P_2}, \cdots]$ (resp. $S_I = [S_{I_1}, S_{I_2}, \cdots]$) approximant le réel a et la suite de points entiers $K_L = [K_{L_1}, K_{L_2}, \cdots]$ (resp. $K_U = [K_{U_1}, K_{U_2}, \cdots]$) approximant la droite L. Les suites de points entiers K_L et K_U forment l'enveloppe convexe des points entiers d'abscisse positive inférieure à b' situés respectivement en dessous et au-dessus de la droite L. Ces suites sont appelées les *voiles de Klein* ou *lignes polygonales de Klein* inférieures et supérieures respectivement (voir [Kle07, Lac98a, Lac98b, Arn98]). Les points entiers déduits des convergents principaux correspondent aux sommets des enveloppes convexes et les points entiers déduits des convergents intermédiaires correspondent à des points portés par les arêtes des enveloppes convexes. La figure 1.11 montre les deux voiles de Klein associées à la droite de pente 4/11 ainsi que la correspondance avec les suites de points entiers K_L et K_U. Dans cet exemple, $K_L = [(1,0),(3,1),(11,4)]$ et $K_U = [(0,1),(1,1),(2,1),(5,2),(8,3),(11,4)]$.

Remarquons que pour calculer l'enveloppe convexe des points entiers différents de l'origine situés dans un cône 2D borné par deux droites de pente rationnelle et d'origine entière O, nous pouvons utiliser les fractions continues. Une telle enveloppe convexe est appelée *polyèdre de Klein*. Supposons sans perte de généralité que les pentes des droites bornant le cône sont positives. Pour construire le polyèdre de Klein associé au cône, il suffit d'appliquer la transformation unimodumaire T qui ramène le côté supérieur du cône au vecteur unitaire vertical $(0,1)$ ou le côté inférieur du cône au vecteur unitaire horizontal $(1,0)$. Nous calculons ensuite les lignes polygonales associées au côté non-horizontal ou non-vertical et en appliquant la transformation inverse T^{-1}, nous retrouvons le polyèdre

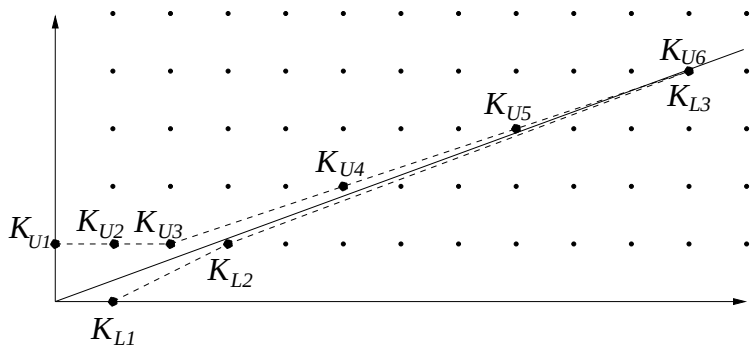


FIG. 1.11 – Exemple de voiles de Klein pour $a = 4/11$

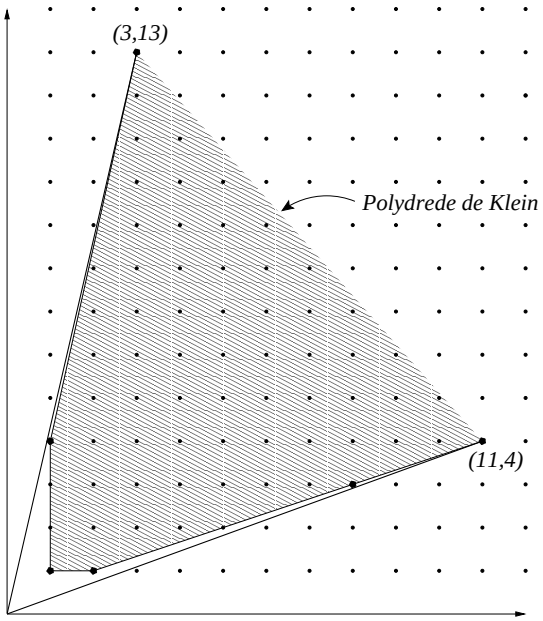


FIG. 1.12 – Exemple de polyèdre de Klein

de Klein du cône. La figure 1.12 montre un exemple de polyèdre de Klein. Pour construire ce polyèdre, nous aurons appliqué à tous les points entiers de la grille la transformation de matrice associée $[13 - 3| - 41]$ afin de ramener le vecteur $(3, 13)$ au vecteur vertical $(0, 1)$. Du coup, le vecteur $(11, 4)$ devient le vecteur $(131, -40)$. Le polyèdre de Klein correspond à la ligne polygonale supérieure de la droite portée par le vecteur $(131, -40)$. Il suffit ensuite d'appliquer la transformation inverse T^{-1}. Pour plus de détails sur les polyèdres de Klein, vous pouvez vous référer à [Mou00].

Nous pouvons interpréter la Proposition 2 dans le plan euclidien.

Proposition 3 *Soit $P = (q, p)$ le point à coordonnées entières situé en dessous (resp. au-dessus) de la droite linéaire L de pente réelle a, tel que la droite passant par P d'équation $y = (p/q)x$ approxime au mieux la droite L et tel que l'abscisse de P n'excède pas un entier donné d. Le point entier P est le terme de la suite K_L (resp. K_U) de a de plus grande abscisse n'excédant pas d.*

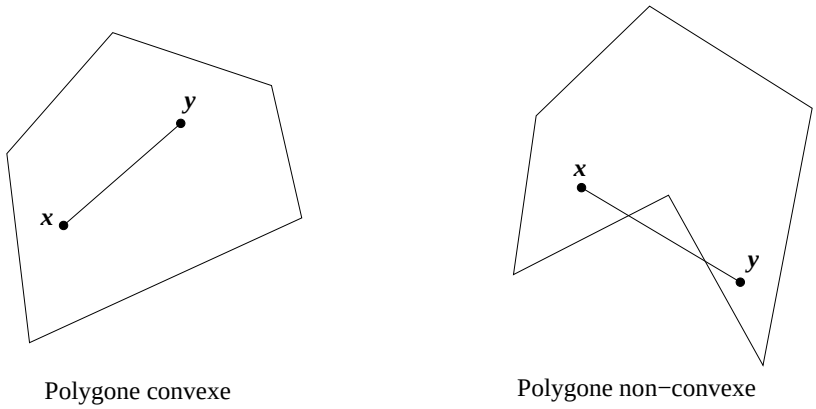


FIG. 1.13 – Exemple de polygone convexe et non-convexe

1.3 Enveloppes Convexes et Algorithmes

Soit A une partie de $\mathbb{R}^n$, de nombreux algorithmes requièrent la détermination de l'enveloppe convexe de A. Dans cette section, nous rappelons la définition des enveloppes convexes et nous introduisons les principaux algorithmes de construction de ces enveloppes en dimension 2. Cet état de l'art est dans la prolongation de celui présenté dans [Buz02].

1.3.1 Définitions

Nous introduisons tout d'abord la notion d'ensembles convexes. Les deux définitions suivantes sont équivalentes.

Définition 6 *Un ensemble C de $\mathbb{R}^n$ est* convexe *si pour tout point x et tout point y de C, le segment xy est entièrement contenu dans C.*

Cette définition peut également être formulée en termes de barycentres.

Définition 7 *Un ensemble C de $\mathbb{R}^n$ est* convexe *si pour toute famille de points pondérés de C de la forme $\{(a_1, \lambda_1), (a_2, \lambda_2), \cdots, (a_s, \lambda_s) : \lambda_i \geq 0, 1 \leq i \leq s\}$ le barycentre de la famille appartient à C.*

La figure 1.13 illustre un exemple de polygone convexe et non-convexe.
Il s'en suit le théorème suivant :

Théorème 2 *L'intersection de deux ensembles convexes est un ensemble convexe.*

Nous pouvons à présent introduire la définition de l'enveloppe convexe d'un ensemble de points. Informellement, l'enveloppe convexe d'un ensemble de points dans le plan pourrait être obtenue en tendant un élastique autour de l'ensemble de points puis en lachant ce même élastique. La forme prise par l'élastique correspond à l'enveloppe convexe de l'ensemble de points.

Définition 8 *L'*enveloppe convexe *d'un ensemble de points A de $\mathbb{R}^n$ est l'ensemble convexe de taille minimale qui contient l'ensemble de points A.*

Elle peut également être définie comme l'intersection de tous les ensembles convexes qui contiennent A. Tout demi-espace affine correspond à un espace convexe, l'enveloppe convexe de l'ensemble de points A correspond donc à l'intersection des demi-espaces affines de $\mathbb{R}^n$ qui contiennent A.

Soit C une partie convexe de $\mathbb{R}^n$, un point x de C est dit *extrémal* s'il n'existe pas de couple de points (y, z) tels que x appartienne au segment ouvert $]y, z[$. L'enveloppe convexe d'un ensemble fini de points correspond à un polytope et ses points extrémaux sont appelés *sommets*. Nous rappelons que le terme polytope correspond à la généralisation des polygones en dimension quelconque et qu'un polytope est toujours borné. Déterminer l'enveloppe convexe d'un ensemble de points fini revient à donc à déterminer l'ensemble de ses sommets. Dans la suite de notre étude, nous nous intéressons plus particulièrement au calcul d'enveloppes convexes dans le plan. L'enveloppe convexe d'un ensemble fini de points dans le plan correspond à un polygone convexe borné.

1.3.2 Panel d'Algorithmes

Nous décrivons dans cette section quelques uns des principaux algorithmes de construction d'enveloppes convexes dans le plan (voir [dBSvKO00, PS85, Ede87, Mul94, O'R98, Buz02] pour des ouvrages traitant de ce sujet). Nous notons A un ensemble de n points de $\mathbb{R}^2$ dont nous souhaitons calculer l'enveloppe convexe. Remarquons que l'approche exhaustive naïve calcule une telle enveloppe convexe en $O(n^4)$. En effet, cette méthode vérifie pour chaque point de l'ensemble A s'il est contenu dans un triangle formé par trois autres points de A. Le nombre de triangles étant borné par $O(n^3)$, sa complexité est en $O(n^4)$. Cependant, il existe de nombreuses autres méthodes, certaines atteignant la complexité optimale en $O(n \log h)$ où h correspond au nombre de sommets de l'enveloppe convexe. De façon générale, les enveloppes convexes construites sont décrites par leurs sommets et ceux-ci sont orientés dans le sens inverse des aiguilles d'une montre.

L'algorithme du Gift Wrapping

La méthode la plus connue se nomme la méthode du *gift wrapping*, ce qui signifie emballage cadeau, également appelée la *marche de Jarvis* [Jar73, CK70]. Cet algorithme doit probablement sa notoriété à son concept qui s'avère assez intuitif. De plus, sa complexité bien que quadratique et donc non-optimale a l'avantage d'être adaptative puisqu'elle s'exprime en fonction du nombre de sommets de l'enveloppe convexe finale. Dans le cas où l'enveloppe convexe finale compte beaucoup moins de sommets que de points dans l'ensemble initial, cette méthode s'avère très rapide. Le principe de l'algorithme, illustré ensuite par la figure 1.14, est le suivant. Il s'agit d'identifier tout d'abord un point de A qui correspond à un sommet de l'enveloppe convexe à calculer. Nous pouvons par exemple choisir le point de plus petite ordonnée et si plusieurs points ont cette même ordonnée, sélectionner celui de plus petite abscisse. Nous imaginons ensuite que nous fixons l'extrémité d'un ruban à ce point, puis que nous faisons tourner le ruban depuis l'horizontale jusqu'à ce qu'il rencontre un autre point de l'ensemble A. Ce nouveau point correspond lui aussi à un sommet de l'enveloppe convexe finale et nous déterminons ainsi une arête de l'enveloppe convexe. En pratique, pour déterminer le sommet c_i à partir des sommets c_{i-2} et c_{i-1} précédemment calculés, nous recherchons le point p de A tel que l'angle entre le vecteur $c_{i-2}c_{i-1}$ et le vecteur $c_{i-1}p$ soit positif et minimal (voir figure 1.14). Cette méthode est résumée dans l'algorithme 3.

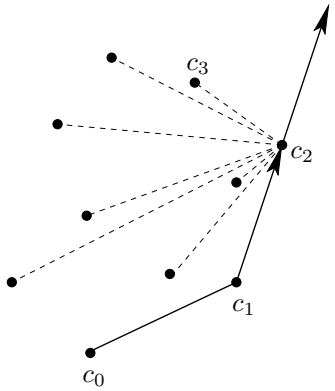


FIG. 1.14 – Exemple d'itération de la méthode du gift wrapping

Algorithme 3 : Algorithme du gift wrapping

GIFT WRAPPING (A : ensemble de n points du plan)
Valeur retournée : sommets de l'enveloppe convexe $C = (c_0, c_1, \cdots)$
1 $c_0 \leftarrow$Point d'ordonnée minimale de A
2 $vecteurRuban \leftarrow (1, 0)$
3 $i \leftarrow 1$
4 FAIRE
5 $\quad c_i \leftarrow$point p de A tel que l'angle $(vecteurRuban, c_{i-1}p)$ est minimal
6 $\quad vecteurRuban \leftarrow c_{i-1}c_i$
7 $\quad i \leftarrow i + 1$
8 TANT QUE $c_{i-1} \neq c_0$

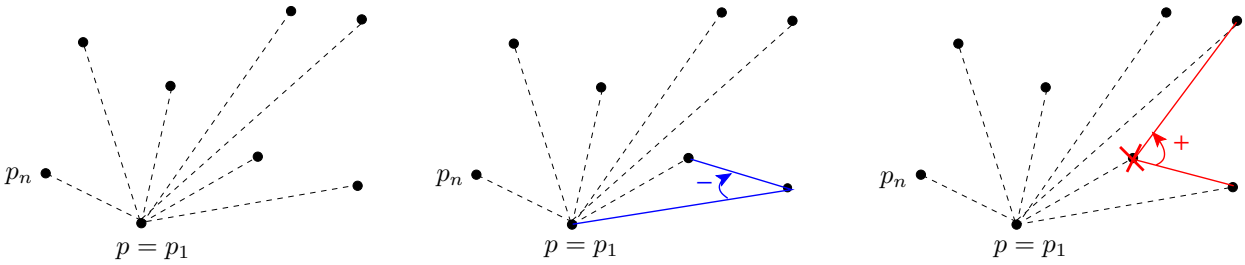


FIG. 1.15 – Exemple du Graham scan. De gauche à droite : calcul des angles par rapport à p, trois points formant un angle négatif et trois points formant un angle positif (élimination du point du milieu)

Notons h le nombre de sommets de l'enveloppe convexe calculée et n le nombre de points de l'ensemble A. À chaque itération, un nouveau sommet de l'enveloppe est calculé en temps $O(n)$. De plus, l'algorithme effectue autant d'itérations que de sommets de l'enveloppe convexe, la complexité globale de l'algorithme du gift wrapping est donc $O(nh)$. Le nombre de sommets de l'enveloppe est borné par le nombre de points de l'ensemble A et dans le pire cas, la complexité de l'algorithme est donc quadratique.

L'algorithme du Graham scan

Le second algorithme que nous avons choisi de décrire a été proposé par Graham dans [Gra72] au début des années 70. Relativement populaire, il est désigné dans la littérature sous le nom de *Graham scan* ou *parcours de Graham*. Cet algorithme fonctionne via une méthode de tri angulaire. La première étape consiste à choisir un point quelconque p de l'ensemble A. Nous pouvons bien sûr considérer comme dans l'algorithme précédent le point de plus petite ordonnée, mais ce n'est pas une obligation. Les points de l'ensemble A sont ensuite triés en fonction de l'angle que chaque point de A et le point p forme avec l'axe des x, dans l'ordre croissant. La suite des points de A ainsi triés forme une étoile autour de P. L'algorithme considère alors tout triplet de points (p_i, p_{i+1}, p_{i+2}) successifs de cette suite. Si ces trois points forment un angle positif, alors le point p_{i+1} est rejeté (voir figure 1.15).

Cette méthode est résumée dans l'algorithme 4. Pour simplifier l'écriture de l'algorithme, nous choisissons comme point de référence pour le tri angulaire le point de A de plus petite ordonnée. Nous utilisons également une pile L et nous notons t l'indice de la tête de la pile. Le point l_t correspond donc à la tête de la pile et le point l_{t-1} correspond au deuxième élément de la pile.

Le tri angulaire effectué au début de l'algorithme se fait en $O(n \log n)$. Reste à évaluer la complexité de la seconde partie de l'algorithme. Nous remarquons tout d'abord que l'ajout et le retrait d'un point dans la pile se font en temps constant. Ainsi, une exécution de la ligne 9 ou de la ligne 10 se fait en $O(1)$. Nous cherchons à évaluer le nombre total d'exécutions des lignes 7 à 10 de l'algorithme. Le test de la ligne 7 est fait $O(n)$ fois ce qui borne le nombre de passages dans la boucle associée. De plus, la ligne 10 est exécutée une fois à chaque passage dans cette même boucle soit également $O(n)$ fois. Cette ligne correspondant à l'ajout d'un point dans la pile, nous savons qu'au plus $O(n)$ points sont ajoutés au total. Nous remarquons que la ligne 9 correspond au retrait d'un point dans la pile. La pile étant initialement vide et sachant qu'au plus $O(n)$ points y sont ajoutés, nous ne pouvons retirer que $O(n)$ points au maximum. La ligne 9 s'exécute donc au plus $O(n)$

Algorithme 4 : Algorithme du Graham scan

```
GRAHAM SCAN (A : ensemble de n points du plan)
Valeur retournée : sommets de l'enveloppe convexe
1  p_1 ←Point d'ordonnée minimale de A
2  Trier angulairement les points de A par rapport à p_1 : (p_2, ··· , p_n)
3  Créer une pile vide L
4  Empiler (p_1)
5  Empiler (p_2)
6  i ← 3
7  TANT QUE i ≤ n
8      TANT QUE angle(l_{t-1}, l_t, p_i) ≥ 0
9          Dépiler l_t
10     Empiler p_i
11     i ← i + 1
12 Retourner les points de la pile L
```

fois. Reste à évaluer le nombre maximum de tests effectués en ligne 8. Nous remarquons que chaque fois que le test est vrai alors un point est retiré de la liste. Le nombre de retraits étant borné en $O(n)$, il en est de même pour le nombre de tests en ligne 8 valant *vrai*. En outre, le test s'avère faux exactement une fois à chaque passage dans la boucle ligne 7, soit $O(n)$ fois. Le nombre maximum de tests est donc borné en $O(n)$ dans le pire cas. Étant donné que la complexité en $O(n \log n)$ du tri prend le pas sur la complexité linéaire du reste de l'algorithme, la méthode du Graham scan s'exécute en $O(n \log n)$.

En 1979, Andrew a proposé une variante de l'algorithme de Graham mettant en oeuvre un tri lexicographique de l'ensemble de points A suivant leurs coordonnées x et y [And79]. Certes, cette méthode impose également l'utilisation d'un algorithme de tri et la complexité de la méthode est inchangée mais les comparaisons angulaires de l'algorithme de Graham sont plus compliquées et surtout plus coûteuses. L'algorithme proposé par Andrew fonctionne de la manière suivante, illustrée sur la figure 1.16. L'algorithme identifie d'abord les deux points de A notés a et b respectivement d'abscisse minimum et maximum. Si plusieurs points ont la même abscisse, l'algorithme considère celui d'ordonnée maximale. L'algorithme construit alors deux chaînes polygonales au-dessus et en dessous de la droite ab. Pour cela, il considère les points de l'ensemble A situés au-dessus (resp. en dessous) de la droite ab et les relie par abscisse croissante afin de former une chaîne polygonale. Si plusieurs points ont la même abscisse, il ne considère que celui d'abscisse maximale (resp. d'abscisse minimale). L'algorithme transforme alors ces deux chaînes polygonales en deux demi-enveloppes convexes. Il suffit de remarquer qu'une suite de trois points (p_1, p_2, p_3) rangés par abscisses croissantes forme un angle négatif si et seulement si le point p_3 est intérieur par rapport à la droite p_1p_2. Cette méthode est donc très similaire au Graham scan mais elle construit à partir des chaînes polygonales supérieures et inférieures allant de a vers b deux demi-enveloppes convexes, l'une orientée dans le sens des aiguilles d'une montre et l'autre orientée dans le sens inverse des aiguilles d'une montre. La figure 1.16 illustre cette méthode sur un exemple.

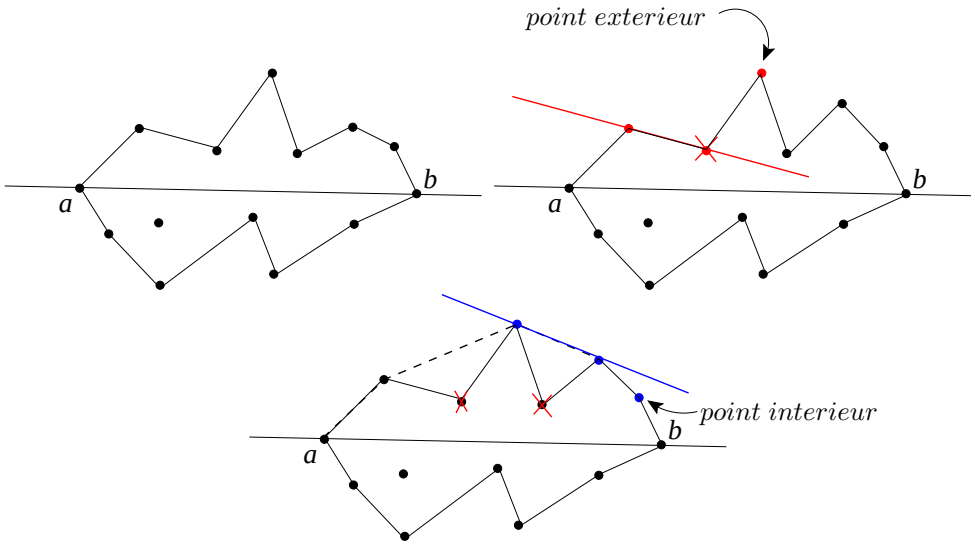


FIG. 1.16 – Exemple de la variante du Graham scan. De gauche à droite : chaînes polygonales, trois points formant un angle positif (élimination du point du milieu) et trois points formant un angle négatif

L'algorithme de Melkman

L'algorithme présenté dans cette section entre dans un cadre un peu différent par rapport aux méthodes décrites précédemment. En effet, au lieu de considérer un ensemble de points quelconque en entrée de l'algorithme de calcul d'enveloppe convexe, nous considèrerons les sommets d'une chaîne polygonale simple. Nous rappelons qu'une chaîne polygonale simple est une chaîne polygonale ne s'intersectant pas avec elle-même. Nous décrivons dans cette section la méthode proposée par Melkman en 1987 dans [Mel87]. Cette approche est algorithmiquement la plus avancée et elle est devenue l'une des principales techniques de calcul d'enveloppe convexe. Elle est illustrée par la figure 1.17.

Soit $S = (s_1, s_2, \cdots, s_n)$ une chaîne polygonale simple de sommets s_i et d'arêtes $s_i s_{i+1}$. L'algorithme construit une liste doublement chaînée de points de S notée L de tête l_t et de queue l_q représentant les sommets de l'enveloppe convexe. L'intérêt d'une liste doublement chaînée est de pouvoir ajouter des éléments en tête et en fin de liste mais également de pouvoir en retirer des éléments en temps constant. Notons que tout au long de l'algorithme, les éléments l_t et l_q correspondent au même point de S et ce point représente de plus le dernier sommet inséré dans l'enveloppe convexe. À chaque itération, l'algorithme décide si le nouveau sommet s_i de S correspond à un sommet de l'enveloppe convexe courante. Si c'est le cas, il insère s_i en tête et en fin de la liste L en éliminant au préalable, si nécessaire, des sommets ajoutés à l'enveloppe convexe durant les itérations précédentes de façon à préserver la convexité.

L'enveloppe convexe est initialisée avec trois sommets consécutifs de la chaîne polygonale S. Nous considèrons ensuite un à un les autres sommets de S. Pour chaque nouveau sommet s_i de S, nous étudions sa position par rapport à $l_t l_{t+1}$ et $l_{q-1} l_q$. Si s_i est intérieur à ces deux arêtes, alors s_i est à l'intérieur de l'enveloppe convexe et il est rejeté. En effet, si s_i était à la fois intérieur à ces arêtes et en dehors de l'enveloppe convexe courante, alors la chaîne polygonale s'auto-intersecterait. Cependant, ceci est impossible par hypothèse. Reste donc à étudier l'autre cas et le comportement de l'algorithme va différer selon la configuration rencontrée. Deux possibilités sont à considérer et peuvent arriver simultané-

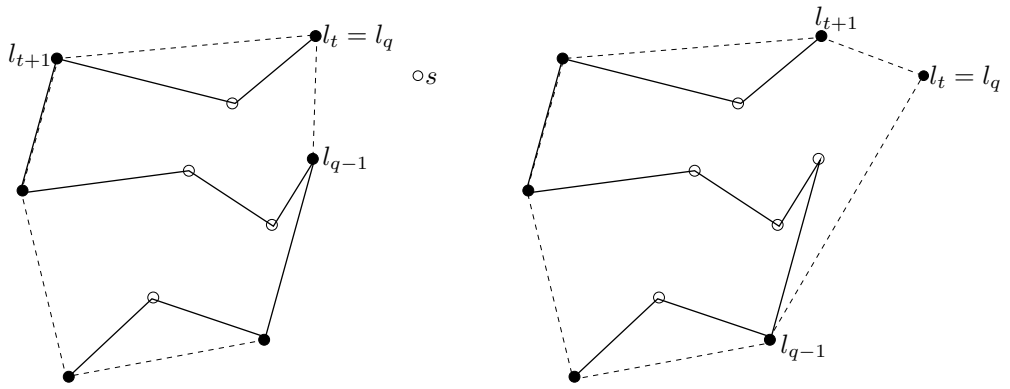


FIG. 1.17 – Exemple de l'algorithme de Melkman : insertion d'un nouveau sommet

ment lors d'une itération. Si s_i est extérieur à $l_t l_{t+1}$, comme dans l'algorithme du Graham scan, nous nous assurons que les sommets $s_i l_t l_{t+1}$ forment un angle négatif en dépilant successivement, si besoin est, la tête de la liste L. Si s_i est extérieur à $l_q l_{q-1}$, nous nous assurons que les sommets $l_{q-1} l_q s_i$ forment un angle négatif en dépilant successivement, si besoin est, la queue de la liste L. Dans tous les cas, nous ajoutons s_i en tête et en fin de la liste L. La figure 1.17 illustre sur un exemple l'insertion d'un nouveau sommet. Cette méthode est résumée dans l'algorithme 5.

Algorithme 5 : Algorithme de Melkman

```
ALGORITHME DE MELKMAN (S : chaîne polygonale à n sommets dans le plan)
Valeur retournée : sommets de l'enveloppe convexe
1  L ←Initialisation avec trois sommets de S
2  POUR TOUT sommet s de S
3    SI s est intérieur à l'enveloppe convexe courante
4      Rejeter s
5    SINON
6      SI s est extérieur à l_t l_{t+1}
7        TANT QUE angle(s l_t l_{t+1}) ≥ 0
8          Dépiler l_t
9      SI s est extérieur à l_q l_{q-1}
10       TANT QUE angle(l_{q-1} l_q s) ≥ 0
11         Dépiler l_q
12     Empiler s en tête et en fin de L
13 Retourner L
```

L'algorithme de Melkman est linéaire suivant le nombre de sommets de la chaîne polygonale. Pour s'en convaincre, nous étudions le nombre maximum d'exécutions de chaque ligne de l'algorithme. Le test de la ligne 2 s'effectue pour chaque point de l'ensemble S soit $O(n)$ fois, ce qui borne le nombre de passages dans cette boucle. Nous avons vu lors de la description de la méthode que pour décider si un point est situé à l'intérieur de l'enveloppe convexe courante en ligne 3 il suffit de tester sa position par rapport à deux arêtes de l'enveloppe. Chacun de ces tests se fait donc en temps constant tout comme les tests des lignes 6 et 9. De plus, ces tests sont effectués au plus une fois à chaque passage dans la boucle de la ligne 2 soit $O(n)$ fois. Comme pour le Graham scan, nous remarquons à présent que l'ajout et le retrait d'un point dans la pile se font en temps constant. En

outre, l'ajout d'un point dans la pile en ligne 12 se fait au plus deux fois à chaque passage dans la boucle de la ligne 2, en tête et en fin de la liste. Le nombre de passages dans cette boucle étant borné en $O(n)$, l'ajout de points est borné en $O(n)$. En ce qui concerne le nombre de retraits de points de la liste en lignes 8 et 11, il est forcément borné par le nombre d'ajouts soit $O(n)$. Reste à évaluer le nombre de tests effectués en lignes 7 et 10. Nous remarquons que chaque fois qu'un de ces tests est vrai alors un point est retiré de la liste. Le nombre de retraits étant borné en $O(n)$, il en est de même pour le nombre de tests en lignes 8 et 11 valant *vrai*. En outre, ces tests s'avèrent faux exactement une fois chacun à chaque passage dans la boucle de la ligne 2, soit $O(n)$ fois. Le nombre maximum de tests est donc borné en $O(n)$ dans le pire cas. Ceci confirme la complexité linéaire de l'algorithme de Melkman suivant le nombre de sommets de la chaîne polygonale.

La Méthode QuickHull

La méthode *QuickHull* (voir [Edd77, Byk78, Gre79, PS85]) doit son nom à l'algorithme de tri QuickSort. En effet, tout comme QuickSort, l'algorithme QuickHull est récursif et consiste à partitionner les données courantes en deux sous-ensembles sur lesquels la méthode est alors relancée. Le partitionnement s'effectue par rapport à trois éléments de l'ensemble courant constituant le *pivot*. Elle procède de manière récursive, un exemple d'itération est illustré par la figure 1.18. Elle détermine d'abord une arête ab de l'enveloppe convexe finale (a d'abscisse inférieure à celle de b) puis recherche le point de l'ensemble A le plus éloigné de cette arête. Ce point c étant extrémal dans une direction, il correspond forcément à un sommet de l'enveloppe convexe. En outre, l'ensemble de points A est entièrement contenu dans la bande définie par la droite ab et par la droite parallèle à ab passant par le point c. L'ensemble de points A est alors séparé en trois sous-ensembles de points disjoints : les points intérieurs au triangle abc, les points extérieurs au triangle associés au segment de droite ac et les points extérieurs au triangle associés au segment de droite bc. Les points du premier sous-ensemble ne peuvent pas correspondre à des sommets de l'enveloppe convexe. L'algorithme est alors relancé recursivement sur les deux autres sous-ensembles. Notons que pour chacun des deux sous-ensembles, une arête de l'enveloppe convexe a déjà été implicitement calculée à l'itération précédente (voir l'exemple en figure 1.18). L'algorithme 6 résume cette méthode.

Algorithme 6 : Algorithme QuickHull

QUICKHULL (A : ensemble de n points du plan,
ab : arête de l'enveloppe convexe de A)
Valeur retournée : sommets de l'enveloppe convexe
1 SI $card(A) = 0$
2 Retourner $\emptyset$
3 $c \leftarrow$Point de A de plus éloigné de ab
4 Partionner A en 3 sous-ensembles : Triangle, Adjacent(ac), Adjacent(bc)
5 Retourner QUICKHULL(Adjacent(ac),ac) $\bigcup$ QUICKHULL(Adjacent(bc),bc)

Cet algorithme s'exécute en $O(nh)$, sa complexité est donc quadratique dans le pire cas. Soit k le nombre de points de l'ensemble considéré lors de l'appel courant de l'algorithme. La détermination du sommet c et le partitionnement de l'ensemble de points se fait en $O(k)$. En effet, pour chaque point de l'ensemble, il est nécessaire de calculer sa distance par rapport à l'arête ab afin d'identifier le point le plus éloigné. Ce point ainsi défini, le

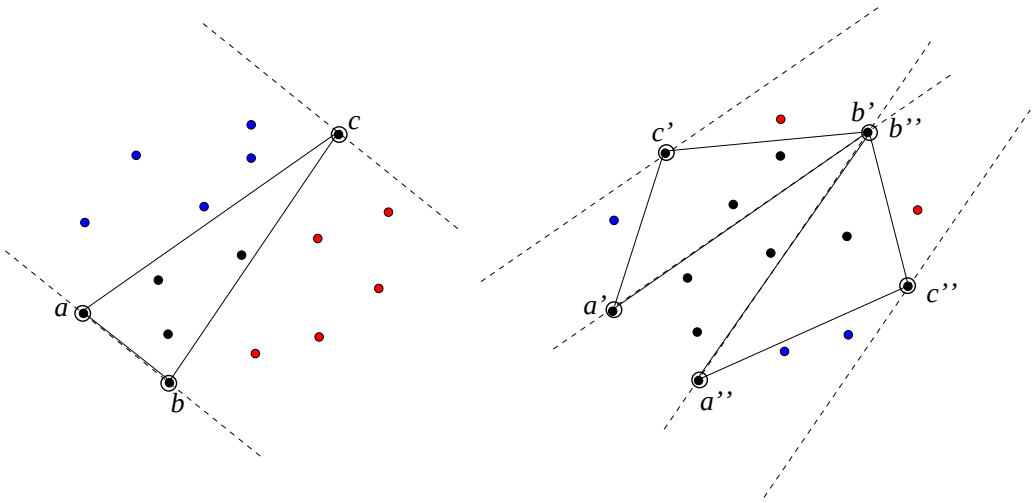


FIG. 1.18 – Exemple de la méthode QuickHull

positionnement de tous les autres points de l'ensemble par rapport aux points a, b et c doit être déterminé afin de partitionner l'ensemble de points. Chaque appel à l'algorithme est donc linéaire en le nombre de points de l'ensemble courant. La pire configuration possible apparaît lorsqu'à chaque appel récursif les points restants se situent tous dans un des deux sous-ensembles Adjacent(ac) ou Adjacent(bc). L'algorithme effectue alors un seul appel récursif sur l'unique sous-ensemble non vide. Remarquons à présent qu'à chaque appel de l'algorithme, nous déterminons un nouveau sommet de l'enveloppe convexe et de ce fait le nombre de points de l'ensemble considéré à l'appel suivant décroît strictement d'au moins 1. De plus, le nombre d'appels récursifs est borné par le nombre de sommets de l'enveloppe convexe finale. Ainsi, en sommant le nombre de points visités au maximum au fil des appels récursifs, nous obtenons une borne dans le pire cas de $O(\sum_{i=0}^{h-1} n - i)$ ce qui revient en termes de complexité à $O(nh)$.

Remarquons enfin que cette méthode peut atteindre une complexité en $O(n \log n)$ dans certaines configurations idéales où les k points de l'ensemble courant sont partitionnés à chaque appel de façon équilibrée entre les sous-ensembles Adjacent(ac) et Adjacent(bc). Dans ce cas, deux appels récursifs sur chacun de ces sous-ensembles sont engendrés, chacun s'exécutant en $O(k/2)$ dans le pire cas. La taille de l'ensemble de points courant étant divisé par deux à chaque nouvel appel, la complexité de l'algorithme s'approche alors de $O(n \log n)$.

La Méthode Diviser pour Régner

L'expression "Diviser pour régner" ou "division/fusion" est bien connue en algorithmique et qualifie des méthodes dont le principe est le suivant : partitionner les données de départ en deux sous-ensembles de même cardinal, résoudre le problème pour chaque sous-ensemble et assembler les deux résultats. Dans le cas de construction d'enveloppes convexes, c'est donc exactement la même chose. Il s'agit de partitionner l'ensemble de points A initial en deux sous-ensembles de points de même cardinal, calculer récursivement leurs enveloppes convexes et calculer l'union des deux enveloppes convexes résultats. Pour partitionner l'ensemble de points initial, nous pouvons nous baser sur le fait que si les points $\{a_1, a_2, \cdots, a_n\}$ de A sont rangés par abscisses croissantes, alors les enveloppes convexes des ensembles $\{a_1, \cdots, a_{n/2}\}$ et $\{a_{(n/2)+1}, \cdots, a_n\}$ sont disjointes. Pour les fu-

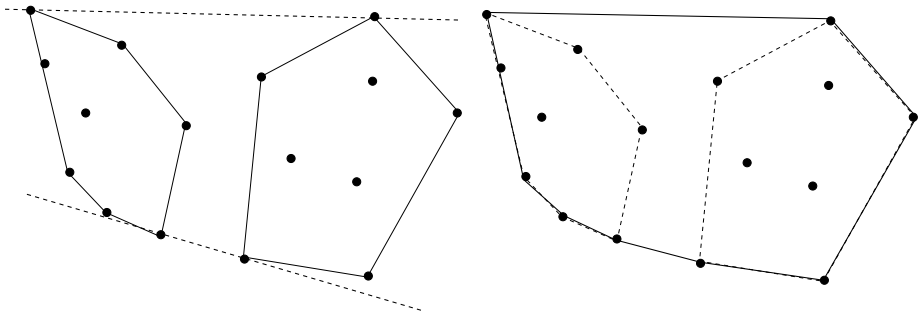


FIG. 1.19 – Exemple de la méthode Diviser pour Régner

sionner, il suffit de calculer leurs deux tangentes communes (voir [PH77] et la figure 1.19 pour l'exemple).

Cette méthode s'exécute en $O(n \log n)$. En effet, nous voyons immédiatement que le nombre d'appels récursifs de la méthode est borné par $O(\log n)$. De plus, à chaque appel, la méthode doit déterminer les deux droites tangentes aux enveloppes convexes. Ce calcul s'effectue en temps linéaire suivant le nombre de sommets des deux enveloppes convexes courantes. Soit $T(n)$ la complexité de la méthode, l'équation de récurrence de l'algorithme est donc : $T(n) = O(n) + 2T(n/2)$. Elle admet pour solution $T(n) = O(n \log n)$.

Algorithme Incrémental

Le principe de l'algorithme incrémental est de construire l'enveloppe convexe de A en considérant ses points au fur et à mesure [Kal84]. Soit A_i l'ensemble des i premiers points de A avec $0 < i < n$ et soit C_i l'enveloppe convexe de A_i. Il s'agit de déterminer la suite d'enveloppes convexes $(C_j)_{1 \leq j \leq n}$, chaque enveloppe convexe C_{i+1} étant construite à partir de C_i comme sur la figure 1.20. Pour chaque nouveau point a_{i+1} considéré, deux possibilités s'offrent à nous. Soit a_{i+1} est à l'intérieur de C_i et il est rejeté, soit il est à l'extérieur de C_i et il correspond à un sommet de C_{i+1}. Dans le deuxième cas, il est nécessaire de déterminer les sommets de C_i visibles depuis le point a_{i+1} comme sur la figure 1.20 (gauche). Les sommets de C_i visibles depuis a_{i+1} sont des sommets consécutifs sur l'enveloppe convexe C_i et s'obtiennent de la manière suivante. Supposons que a_{i+1} soit situé à droite de l'enveloppe convexe courante. Soit T_L et T_U les deux droites tangentes à C_i passant par a_{i+1} situées respectivement en dessous et au-dessus de C_i. Notons respectivement g et d les points d'intersection de l'enveloppe convexe courante C_i avec les tangentes T_L et T_U. La suite des sommets de C_i visibles depuis a_{i+1} est alors notée $(c_0, c_1, \cdots, c_j)$ avec $c_0 = g$ et $c_j = d$. Sur l'exemple de la figure 1.20 (gauche), les sommets visibles apparaissent en rouge. Le nouveau point a_{i+1} est alors ajouté à C_{i+1} entre g et d tandis que les sommets $(c_1, \cdots, c_{j-1})$ sont retirés de l'enveloppe convexe (voir la figure 1.20).

Étudions la complexité de cette méthode résumée par l'algorithme 7. Pour chaque nouveau point considéré, l'algorithme doit d'abord décider s'il se situe à l'intérieur ou à l'extérieur de l'enveloppe convexe courante. Ceci se fait en temps linéaire suivant le nombre de sommets de l'enveloppe convexe courante. Or, exactement un nouveau sommet est ajouté à l'enveloppe convexe à chaque itération. Pour l'ensemble des points de A, cette vérification se fait donc en $O(n^2)$. Si le point est à l'intérieur de l'enveloppe convexe courante, l'algorithme ne le traite pas et ne le considère plus jusqu'à la fin de l'exécution. Si le point se situe à l'extérieur, l'algorithme recherche les sommets de l'enveloppe convexe courante visibles depuis ce point. Les sommets visibles étant consécutifs sur l'enveloppe

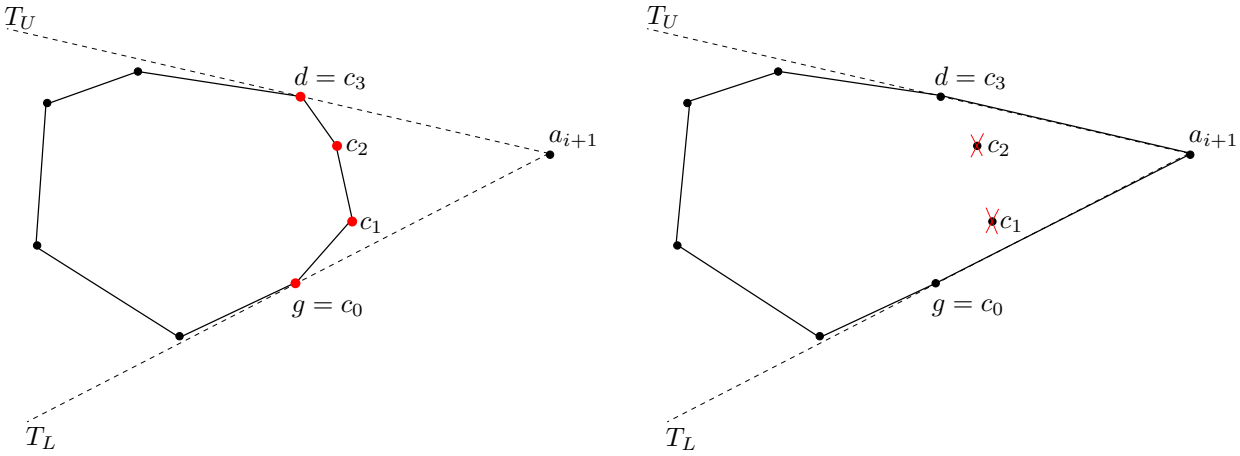


FIG. 1.20 – Exemple de la méthode incrémentale

Algorithme 7 : Algorithme de calcul d'enveloppe convexe incrémental

ENVELOPPE CONVEXE INCREMENTAL (A : ensemble de n points du plan)
Valeur retournée : sommets de l'enveloppe convexe
1 $i \leftarrow 3$
2 $C_i \leftarrow (a_1, a_2, a_3)$
3 POUR i allant de 4 à n FAIRE
4 SI a_i à l'extérieur de C_{i-1}
5 Trouver les arêtes visibles de C_{i-1} depuis a_i
Soient g et d les sommets extrémaux de la portion visible
6 Supprimer ces arêtes
7 Insérer a_i entre g et d dans la liste représentant C_i
8 Retourner C_n

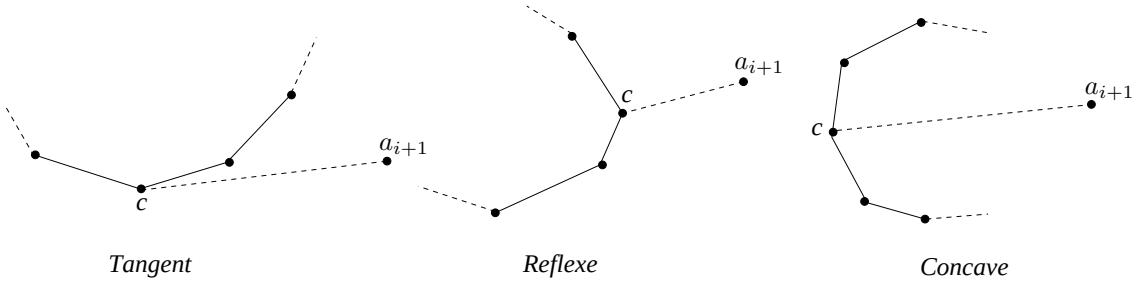


FIG. 1.21 – Les trois configurations du segment $a_{i+1}c$

convexe, une fois qu'un premier sommet visible est trouvé, un parcours vers la droite et vers la gauche le long de l'enveloppe convexe permet de déterminer les autres. La détermination d'un premier sommet visible se fait en temps linéaire suivant le nombre de sommets de l'enveloppe convexe courante. La recherche des sommets visibles se faisant dans le pire cas pour chaque point de l'ensemble A, cette recherche se fait donc en $O(n^2)$ au total. Par conséquent, la complexité de cet algorithme incrémental est en $O(n^2)$.

De toute évidence, l'opération dont la complexité doit être améliorée correspond à la détection des sommets visibles. Plus précisément, nous souhaitons déterminer en temps logarithmique les points d'intersection des deux droites tangentes à l'enveloppe convexe passant par le point a_{i+1} à insérer, notés précédemment g et d. Si nous nous positionnons au point a_{i+1} à insérer en "regardant" dans la direction de l'enveloppe convexe, le point g correspond au point le plus à gauche et le point d correspond au point le plus à droite (voir la figure 1.20). Preparata introduit dans [Pre79] une méthode logarithmique pour déterminer ces deux points. Une description détaillée de cet algorithme est donnée dans [PS85]. Le principe général de cette méthode est d'utiliser un arbre de recherche équilibré dont les noeuds correspondent aux sommets de l'enveloppe convexe courante. En faisant une recherche dichotomique sur l'arbre, nous parvenons à déterminer d'une part le sommet g et d'autre part le sommet d. Soient a_{i+1} le point à insérer comme sommet de l'enveloppe convexe et c un sommet de cette enveloppe, nous distinguons trois configurations possibles : le segment $a_{i+1}c$ est tangent à l'enveloppe convexe, le segment $a_{i+1}c$ est réflexe par rapport à l'enveloppe convexe ou le segment $a_{i+1}c$ est concave par rapport à l'enveloppe convexe. Ces trois configurations sont illustrées sur la figure 1.21.

Selon la configuration rencontrée, la recherche du sommet g ou du sommet d est orientée différemment. L'algorithme 8 résume la méthode utilisée pour rechercher le sommet g. Notons que cet algorithme peut facilement être adapté pour rechercher le sommet d. De plus, dans le cas où le point a_{i+1} est situé à l'intérieur de l'enveloppe convexe courante, les sommets g et d n'existent pas et l'algorithme renvoie l'ensemble vide. La figure 1.22 illustre un exemple de détermination du sommet g lors de l'ajout d'un nouveau sommet a_{i+1}. Dans cet exemple, les sommets de l'enveloppe convexe courante sont notés c_1, c_2, $\cdots$, c_8 et l'arbre de recherche correspondant a pour racine c_4. Le sommet c_4 est réflexe par rapport à a_{i+1}, ce qui oriente la recherche vers la partie gauche de l'arbre. L'algorithme considère alors le sommet c_1, concave par rapport à a_{i+1}, et le sommet c_2 est alors sélectionné. Ce dernier sommet est tangent et correspond donc au sommet g recherché.

En pratique, l'algorithme de recherche s'avère un peu plus compliqué car la détermination du sous-arbre à considérer à l'itération suivante n'est pas triviale. Preparata indique qu'en pratique, si le segment $a_{i+1}c$ n'est pas tangent à l'enveloppe convexe courante, huit configurations sont à considérer au lieu des deux présentées en figure 8. Les lecteurs sont

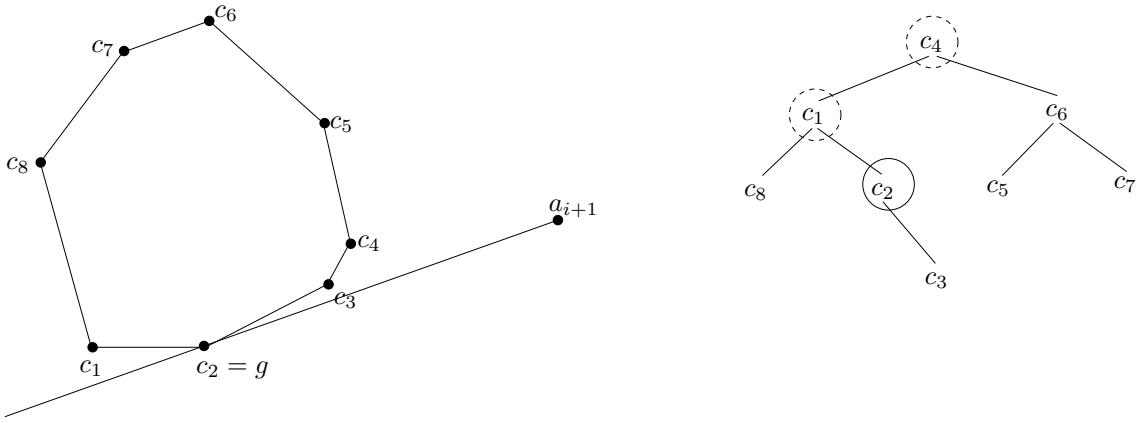


FIG. 1.22 – Exemple de détermination du sommet g

Algorithme 8 : Algorithme de recherche du sommet g dans l'arbre

RECHERCHE-GAUCHE (T : arbre des sommets de l'enveloppe convexe,
a_{i+1} : sommet à ajouter)
Valeur retournée : sommet g
1 $c \leftarrow$Racine(T)
2 SI T est vide
3 Retourner $\emptyset$
4 SI $a_{i+1}c$ est tangent
5 Retourner c
6 SI $a_{i+1}c$ est réflexe
7 $T \leftarrow$ ArbreGauche(c)
8 SINON
9 $T \leftarrow$ ArbreDroit(c)
10 RECHERCHE-GAUCHE(T)

invités à se référer à [PS85] pour de plus amples détails.

Edelsbrunner propose une autre façon d'atteindre la complexité en $O(n \log n)$ pour le calcul d'enveloppe convexe incrémental en 2D dans [Ede87]. Il a l'idée de trier au préalable les points de l'ensemble A par abscisses croissantes. De cette façon, tout nouveau point considéré est forcément situé à l'extérieur de l'enveloppe convexe courante ce qui évite des tests. Il remarque de plus que le sommet de C_i le plus proche de a_{i+1} suivant l'axe des x est forcément visible depuis a_{i+1}. En connaissant un sommet visible, il est très facile ensuite de déterminer tous les autres en visitant les sommets de l'enveloppe convexe courante de part et d'autre du sommet initial. Par conséquent, la détermination des sommets visibles se fait en $O(n)$ au total. En effet, les sommets visités sont supprimés et ne seront donc plus visités à nouveau. La complexité de la méthode de calcul d'enveloppe convexe est donc en $O(n \log n)$ à cause du tri. De plus, même si elle requiert un tri sur les points, elle a l'avantage d'être beaucoup plus simple que la méthode de Preparata.

En Dimensions 3

Dans ce mémoire, nous nous intéressons principalement aux algorithmes de construction d'enveloppes convexes dans le plan. Cependant, avant de conclure ce chapitre, nous proposons un rapide tour d'horizon des méthodes existantes pour la construction d'enveloppes convexes en dimension 3 (voir [dBSvKO00] pour une étude plus détaillée).

L'enveloppe convexe d'un ensemble de n points de $\mathbb{R}^3$ est un polytope convexe dont les sommets correspondent à des points de l'ensemble de départ. Il s'agit donc de calculer ces sommets mais aussi les faces du polytope ; ceci constitue un problème nettement plus difficile que dans le plan. Pour le résoudre, différentes méthodes existent dans la littérature. Le premier algorithme de calcul d'enveloppes convexes 3D est une variante de l'algorithme du *gift wrapping* dûe à Chand et Kapur [CK70] dont la complexité est en $O(nf)$ avec n le nombre de points de l'ensemble de départ et f le nombre de faces du polytope résultat. Le nombre de faces de l'enveloppe convexe étant borné par $6n - 20$ [dBSvKO00], cette complexité est de l'ordre de $O(n^2)$ dans le pire cas. L'algorithme *diviser pour régner* peut également se généraliser en dimension 3 [PH77, PS85] atteignant alors une complexité de $O(n \log n)$. C'est également le cas pour l'algorithme *quickhull* [BDH96] s'exécutant en moyenne en $O(n \log n)$ et en $O(n^2)$ dans le pire cas. Notons que Clarkson et Shor proposent dans [CS89] un algorithme incrémental randomisé pour le calcul d'enveloppes convexes 3D basé sur un graphe de conflits et atteignant une complexité de $O(n \log n)$ (voir également [dBSvKO00, Mul94]). Ce dernier algorithme se généralise également aux dimensions supérieures, il est de plus optimal dans le pire cas, s'exécutant en temps $\Theta(n^{\lfloor d/2 \rfloor})$ avec d la dimension de l'espace considérée. Le meilleur algorithme déterministe connu à ce jour en dimension impaire quelconque est d'après [dBSvKO00] une version dérandomisée de cette dernière méthode [Cha93]. De plus, le meilleur algorithme adaptatif a été proposé dans [Cha96] et atteint une complexité en $O(n \log k + (nk)^{1-1/(\lfloor d/2 \rfloor + 1)} \log^{O(1)})$ où k représente la "complexité" de l'enveloppe convexe, c'est-à-dire l'évaluation du nombre de ses sommets, arêtes et faces.

Conclusion

Parmi toutes les méthodes de calcul d'enveloppes convexes planaires exposées dans cette section, la seule qui a la propriété d'être adaptative est la marche de Jarvis. En effet, sa complexité en $O(nh)$, avec n le nombre de points de l'ensemble et h le nombre de

sommets de l'enveloppe convexe à calculer, s'exprime par rapport au nombre de sommets de l'enveloppe convexe finale. Cependant, cette complexité n'est pas optimale puisque l'algorithme s'avère quadratique dans le cas où $n = h$. Les autres méthodes atteignent des complexités plus intéressantes puisqu'elles sont en $O(n \log n)$ mais elles ne sont pas adaptatives. En 1986, Kirkpatrick et al. proposèrent le premier algorithme en $O(n \log h)$ [KS86]. Cette méthode, parfois appelée *Marriage before Conquest*, se base sur l'algorithme Diviser pour Régner mais calcule d'abord les tangentes des enveloppes convexes afin de ne construire effectivement que les parties des enveloppes qui seront conservées. En 1996, Chan proposa dans [Cha96] une autre méthode combinant à la fois la méthode du Graham scan et la marche de Jarvis pour obtenir lui aussi un algorithme optimal et adaptatif en $O(n \log h)$. Le tableau 1.1 liste les algorithmes présentés dans cette section ainsi que leur complexité.

Algorithme	Complexité	Références
Exhaustif	$O(n^4)$	
Marche de Jarvis	$O(nh)$	[Jar73, CK70]
Graham scan	$O(n \log n)$	[Gra72]
QuickHull	$O(nh)$	[Edd77, Byk78]
Diviser pour Régner	$O(n \log n)$	[PH77]
Incrémental	$O(n \log n)$	[Pre79, Ede87]
Marriage before Conquest	$O(n \log h)$	[KS86]
Combinaison Graham/Jarvis	$O(n \log h)$	[Cha96]
Algorithme de Melkman (chaînes polygonales)	$O(n)$	[Mel87]

Tab. 1.1 – Comparatif des algorithmes de construction d'enveloppes convexes dans le plan

1.4 Dénombrement

1.4.1 Théorème de Pick

Un théorème très utile pour calculer l'aire d'un polygone dont les sommets sont portés par la grille est dû à Georg Alexander Pick en 1899 et s'appelle le *théorème de Pick* [Pic99] :

Théorème 3 *Soit P un polygone à sommets portés par une grille régulière, soit I le nombre de points entiers à l'intérieur du polygone et soit B le nombre de points sur les bords du polygone. L'aire A du polygone P peut être exprimée en fonction de I et de B de la manière suivante :*

$$A = I + \frac{B}{2} - 1$$

Ce théorème est valide pour tout polygone simple, c'est-à-dire pour tout polygone constitué d'une seule pièce et sans trou. Il est illustré par l'exemple 1. Ce théorème peut être démontré par induction de la manière suivante. Nous définissons un *polygone élémentaire* comme un polygone sans point entier à l'intérieur et dont les points sur les bords se résument à ses sommets. Il s'agit de montrer tout d'abord que tout triangle élémentaire est

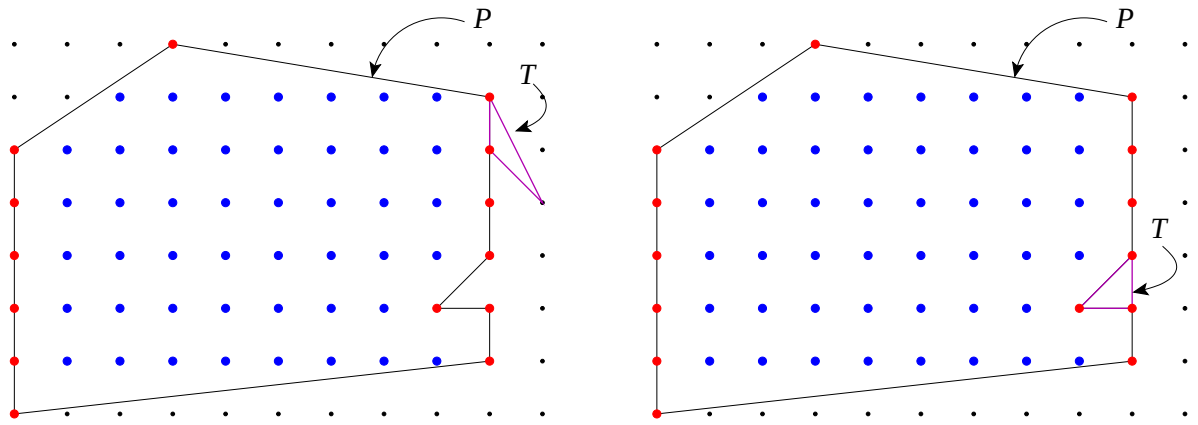


FIG. 1.23 – Ajout d'un triangle élémentaire

d'aire 1/2. Nous remarquons ensuite que tout polygone correspond à l'union d'un nombre fini de triangles élémentaires. Soit P un polygone vérifiant le théorème de Pick et T un triangle élémentaire, nous montrons alors que le polygone $P \cup T$ vérifie aussi le théorème de Pick et nous prouvons donc par induction le théorème.

Nous détaillons à présent la preuve du théorème de Pick. Nous remarquons que tous vecteurs u et v générant un parallélogramme élémentaire forment une base. Ceci signifie, comme expliqué dans la section 1.1.1, que tous les points de $\mathbb{Z}^2$ peuvent être exprimés comme combinaison linéaire à coefficients entiers de u et v. D'après la proposition 1, nous savons que toute base (u, v) peut être obtenue en appliquant un nombre fini de transvections sur les vecteurs de la base orthonormée $((1, 0), (0, 1))$. Comme les transvections sont inversibles, cela signifie que tous vecteurs u et v formant une base peuvent être ramenés par transformations inverses aux vecteurs $(1, 0)$ et $(0, 1)$. Tout parallélogramme élémentaire peut donc être transformé par transvection en un carré de côté 1. D'après la propriété 1, les transvections conservent les aires et tout parallélogramme élémentaire est donc d'aire 1. Nous remarquons que tout parallélogramme élémentaire peut être coupé en deux triangles élémentaires de même aire suivant la diagonale. Nous en déduisons que tout triangle élémentaire est d'aire 1/2. Soit P un polygone comportant I points entiers intérieurs et B points entiers sur les bords. Nous supposons que P vérifie le théorème de Pick et donc que son aire A vérifie $A = I + B/2 - 1$. Soit T un triangle élémentaire, nous montrons alors que l'aire A' de $P \cup T$ vérifie bien $A' = I + B/2 - 1 + 1/2 = I + B/2 - 1/2$. Pour cela, il suffit de remarquer que le triangle élémentaire T peut être assemblé de deux manières différentes avec P : soit P et T ont un côté en commun, soit P et T ont deux côtés en commun. Dans le cas où P et T ont un seul côté en commun (figure 1.23 gauche), le polygone $P \cup T$ a le même nombre de points intérieurs que P mais un sommet en plus, donc un point sur les bords en plus, et $A' = I + (B + 1)/2 - 1 = I + B/2 - 1/2$. Dans le cas où P et T ont deux côtés en commun (figure 1.23 droite), le polygone $P \cup T$ a un point intérieur en plus par rapport à P et un sommet en moins, donc un point sur les bords en moins, et $A' = (I + 1) + (B - 1)/2 - 1 = I + B/2 - 1/2$. Tout polygone pouvant être décomposé en un nombre fini de triangles élémentaires et tout triangle élémentaire étant d'aire 1/2, le théorème de Pick est donc prouvé par récurrence pour tout polygone.

Exemple 1 *Prenons l'exemple du polygone P de la figure 1.24. Ce polygone à sommets portés par la grille contient exactement* 56 *points indiqués en bleu ($I = 56$) et* 16 *points sur ses bords, indiqués en rouge ($B = 16$). Son aire est donc égale à* $16 + (56/2) - 1$ *soit* 43.

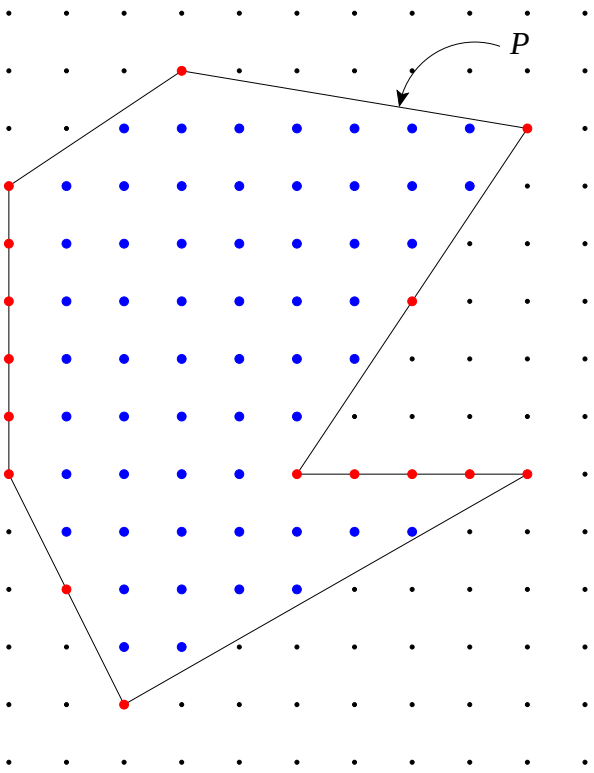


FIG. 1.24 – Application du théorème de Pick

Ce théorème peut par exemple être utilisé pour décider si un polygone dont nous connaissons les sommets contient des points entiers. À partir des coordonnées de ses sommets, nous pouvons calculer de manière classique son aire. Ceci se fait en calculant la somme des aires des triangles contenus dans le polygone. De plus, nous pouvons calculer le nombre de points sur ses bords. En connaissant l'aire du polygone et le nombre de points sur ses bords, nous en déduisons le nombre de points intérieurs.

1.4.2 Le polytope du *Problème du Sac à Dos*

Le *problème du sac à dos*, également connu sous le nom de *knapsack problem*, est un problème bien connu de programmation linéaire en nombres entiers, proposé pour la première fois par Mathews dans [Mat97]. Il s'agit de remplir un sac à dos avec des objets dont nous connaissons le poids et la valeur. Le sac à dos ne pouvant supporter plus d'un certain poids, il convient donc de faire les bons choix afin de maximiser la valeur totale du sac à dos sans dépasser le poids maximum. Évidemment, les objets ne peuvent pas être "coupés en deux" mais nous pouvons prendre plusieurs exemplaires d'un même objet. Nous nous retrouvons alors clairement avec un problème de programmation linéaire en nombres entiers qui peut être formalisé de la manière suivante :

$$\begin{array}{ll} \text{maximiser} & c_1x_1 + c_2x_2 + \cdots + c_dx_d \\ \text{sous la contrainte} & a_1x_1 + a_2x_2 + \cdots + a_dx_d \leq b \\ \text{avec} & x \geq 0 \end{array} \tag{1.2}$$

où a et c correspondent à des vecteurs de $(\mathbb{Z}^+)^d$ et b correspond à un entier positif. Dans cette formulation, l'entier b correspond au poids total pouvant être supporté par le sac à dos. De plus, à chaque objet i, $1 \leq i \leq d$, est associé une valeur c_i représentant l'importance accordée à cet objet, un poids a_i et enfin une quantité x_i. Nous appelons

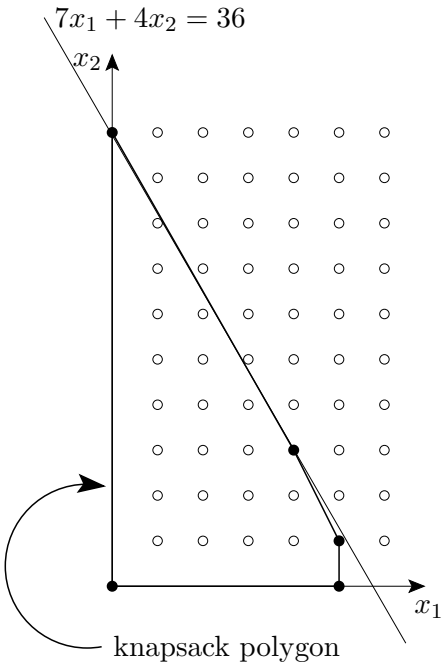


FIG. 1.25 – Polygone du problème du sac à dos (knapsack polygon)

solution réalisable du problème du sac à dos tout point de $(\mathbb{Z}^+)^d$ satisfaisant l'inéquation de (1.2). Le *polytope du problème du sac à dos* (knapsack polytope), noté K, correspond à l'enveloppe convexe de l'ensemble des solutions réalisables du problème du sac à dos et s'écrit :

$$K = \text{conv}\{x = (x_1, \cdots, x_d) \in (\mathbb{Z}^+)^d : a_1x_1 + a_2x_2 + \cdots + a_dx_d \leq b\}$$

Parmi les solutions réalisables, la solution optimale maximisant la valeur totale du sac à dos $c_1x_1 + c_2x_2 + \cdots + c_dx_d$ est recherchée. Il est prouvé qu'elle est toujours atteinte pour un sommet du polytope du problème du sac à dos ([Ste01, Zol00]). En dimension 2, nous considèrons que nous ne disposons que de deux types d'objets à mettre dans le sac à dos. Le polytope du problème du sac à dos correspond alors à un polygone convexe à sommets portés par la grille comme sur l'exemple de la figure 1.25. Tout point entier de coordonnées (x, y) inclus dans ce polygone représente la possiblité de remplir le sac à dos avec x objets du premier type et y objets du second type, sans dépasser le poids maximum autorisé.

Soit V l'ensemble des sommets du polytope du problème du sac à dos. Le cardinal de V s'avère relativement petit et plusieurs études cherchent à borner cette quantité. Soit $\gamma = min\{a_1, \cdots, a_d\}$, Hayes et al. montrent dans [HL83] qu'il admet une borne supérieure et que :

$$|V| < (\log_2((4b)/\gamma))^d) \tag{1.3}$$

C'est en 2000 que Zolotykh améliore cette borne et prouve dans [Zol00] que :

$$|V| \leq d\log(2d)(1 + \log(1 + b/\gamma))^{d-1} \tag{1.4}$$

D'après ce dernier résultat, nous pouvons conclure que lorsque la dimension d est fixée, le cardinal de V est borné par $O(\log(b/\gamma)^{d-1})$. Dans le chapitre suivant, nous mettons en place un algorithme calculant l'enveloppe convexe des points entiers de la grille bordant

une droite et situés dans un domaine vertical borné. Le résultat de Zolotykh nous sera alors utile afin de borner le nombre de sommets de l'enveloppe convexe à calculer.

Chapitre 2

Approximation d'un Nombre Réel par un Rationnel à Dénominateur Borné

2.1 Introduction

Nous considérons le problème suivant : déterminer le nombre rationnel à dénominateur borné qui approxime au mieux un nombre réel a donné. Plus formellement, il s'agit de trouver le nombre rationnel de la forme p/q qui approxime au mieux le réel a tel que son dénominateur q appartienne à l'intervalle borné $[b, b']$ avec b et b' des entiers positifs tels que $b \leq b'$. Le problème géométrique correspondant consiste à déterminer la droite L' de pente rationnelle p/q passant par l'origine qui approxime au mieux la droite L de pente réelle a passant par l'origine relativement à un domaine vertical D de la forme $\{(x, y) \in \mathbb{R}^2 | b \leq x \leq b'\}$. La droite recherchée passe par le point à coordonnées entières (q, p) appartenant au domaine vertical D tel qu'aucun point à coordonnées entières du domaine D ne se situe strictement entre la droite L et la droite L'. Il s'agit donc de trouver la droite L' passant par un point à coordonnées entières du domaine D qui minimise la valeur absolue de l'angle entre les droites L et L'. Nous remarquons que le point (q, p) recherché correspond à un sommet des enveloppes convexes des points de la grille situés au-dessus ou en dessous de la droite L et appartenant au domaine vertical D. Par conséquent, si nous connaissons les sommets de ces deux enveloppes, nous pouvons facilement déterminer la droite approximée que nous recherchons. Dans la littérature, différents algorithmes généraux de calculs d'enveloppes convexes existent (voir la section 1.3.2). Soit h le nombre de sommets des enveloppes convexes à calculer. Nous avons présenté dans la section 1.3.2 un panel des algorithmes les plus connus pour le calcul d'enveloppes convexes. Le plus célèbre s'appelle l'algorithme du *gift wrapping*. Cette méthode calculerait les deux enveloppes convexes en $O((b' - b)h)$ tandis que Chan a proposé en 1996 un algorithme de complexité optimale en temps qui calculerait les enveloppes en $O((b' - b) \log h)$ (voir [Jar73, Cha96]). Cependant, dans notre configuration un peu particulière, d'autres algorithmes atteignent une meilleure complexité. Nous nous intéressons en particulier à deux approches pour calculer ces enveloppes convexes. La première méthode utilise la théorie des nombres et plus précisément l'interprétation géométrique de la décomposition en fractions continues de a, appelée voiles de Klein (voir la section 1.2.2). Cette méthode est similaire à celle proposée par Harvey dans [Har99] et s'exécute en $O(\log(b'))$. Cette approche nécessite de connaître un point à coordonnées entières porté par la droite à

approximer et de s'en servir comme origine du repère. Cependant, cette complexité peut être améliorée surtout si b' est beaucoup plus grand que la largeur du domaine D. Notons que d'autres méthodes basées sur la théorie des nombres pourraient peut être déterminer cette approximation sans la connaissance a priori d'un point à coordonnées entières porté par la droite en utilisant le système de numération d'Ostrowski à la manière de [BI09]. La seconde méthode utilise la géométrie algorithmique comme l'algorithme proposé par Balza-Gomez et al. dans [BGMM99] qui atteint une complexité en temps de $O(\log(b'-b))$. Nous proposons une nouvelle approche pour calculer ces deux enveloppes convexes (voir [CB08c, CB08b, CB09]). Notre méthode combine à la fois la théorie des nombres et la géométrie algorithmique. Elle préserve la complexité optimale en temps de la méthode de Balza-Gomez et al. Notons de plus qu'elle est la première dont la complexité dépend également de la taille de la sortie. Un tel algorithme est qualifié de *output sensitive* ou algorithme adaptatif. En effet, notre méthode calcule un sommet d'une des deux enveloppes convexes à chaque itération. Chaque itération s'exécutant en temps constant, notre méthode a une complexité en $O(h)$ où h représente le nombre de sommets des deux enveloppes convexes construites. De plus, l'algorithme résultant est plus simple et s'avère donc plus adapté pour l'implémentation. Les résultats présentés dans ce chapitre ont par ailleurs été publiés dans [CB08c, CB08b, CB09].

Ce chapitre s'organise de la manière suivante. Nous commençons par expliquer dans la section 2.2 pourquoi le problème d'approximation dans le plan euclidien peut être résolu en calculant des enveloppes convexes. Ensuite, nous introduisons la première approche utilisant la décomposition en fractions continues. Nous nous intéressons tout d'abord à un sous-problème qui consiste à déterminer la meilleure approximation rationnelle d'un nombre réel telle que son dénominateur admette une borne supérieure. Nous montrons dans la section 2.3 comment résoudre numériquement et géométriquement ce problème en utilisant la décomposition en fractions continues. Par la suite, nous traitons d'un problème d'approximation plus contraint : déterminer la meilleure approximation rationnelle d'un nombre réel telle que son dénominateur admette à la fois une borne inférieure et une borne supérieure entière. Nous nous intéressons à l'interprétation géométrique de ce problème et nous montrons dans la section 2.4 comment Harvey résoud ce problème en utilisant les fractions continues ([Har99]). Nous introduisons dans la section 2.5 la seconde approche mettant en oeuvre la géométrie algorithmique pour résoudre la version géométrique de notre problème d'approximation. À cette occasion, nous décrivons l'algorithme de Balza-Gomez et al. ([BGMM99]). Nous présentons dans la section 2.6 notre nouvel algorithme pour la construction des enveloppes convexes et nous étudions sa complexité en temps. Enfin, nous mettons en évidence dans la section 2.7 des applications pour lesquelles notre algorithme peut s'avérer utile.

2.2 Droite Approximée et Enveloppe Convexe Entière

Soit a un nombre réel positif et soit L une droite d'équation $y = ax$. Nous supposons sans perte de généralité que a est inférieur à 1. Soient b et b' deux nombres entiers positifs tels que b est inférieur à b'. Soit D le domaine vertical de la forme $\{(x, y) \in \mathbb{R}^2 | b \leq x \leq b'\}$. Dans la partie géométrique de ce chapitre, nous parlons souvent de points à coordonnées entières. Dans un souci de clarté, nous utilisons donc le terme "point entier" plutôt que le

terme “point à coordonnées entières”. De la même manière, nous utilisons le terme “enveloppe convexe entière” pour décrire l'enveloppe convexe d'un ensemble de points entiers, les sommets d'une telle enveloppe étant eux-même entiers.

Nous nous intéressons au problème de détermination du nombre rationnel p/q qui approxime au mieux le nombre réel a tel que son dénominateur q appartienne à l'intervalle borné $[b, b']$.

Nous établissons une correspondance entre le nombre rationnel p/q et le point entier (q, p) dans le plan euclidien. En effet, toute droite de pente rationnelle d'équation $y = (p/q)x$ passe par le point entier (q, p). Par conséquent, le problème de l'approximation par un rationnel est équivalent, dans le plan euclidien, à trouver le point entier $P = (q, p)$ appartenant au domaine D tel que la droite passant par l'origine et par le point P approxime au mieux la droite d'équation $y = ax$ par rapport au domaine D. Ce point $P = (q, p)$ est appelé *meilleur point approximant* la droite L par rapport au domaine D lorsque P vérifie les conditions suivantes : P appartient au domaine D et la valeur absolue de l'angle entre les deux droites de la forme $y = ax$ et $y = (p/q)x$ est minimal. Ceci implique qu'aucun point entier du domaine D ne se situe entre ces deux droites.

Le problème d'approximation dans le plan euclidien est lié au calcul d'enveloppes convexes entières. Notons CH_U et CH_L les enveloppes convexes des points entiers du domaine D situés respectivement au-dessus et en dessous de la droite L (voir la figure 2.1). De toute évidence, le meilleur point approximant P que nous cherchons à déterminer correspond à un des sommets de CH_U ou de CH_L. En effet, quel que soit le point entier (r, s) dans le domaine D n'appartenant pas aux enveloppes convexes, il existe un sommet de CH_U ou de CH_L situé entre les deux droites d'équations $y = ax$ et $y = (s/r)x$. Notons h le nombre de sommets des enveloppes convexes CH_U et CH_L. D'après [Zol00], résultat présenté en section 1.4.2, nous savons que h est logarithmique en la largeur du domaine D, i.e. CH_U et CH_L comportent $O(\log(b'-b))$ sommets. Ainsi, si nous avons calculé ces deux enveloppes convexes, nous pouvons résoudre le problème d'approximation en $O(h)$. Pour calculer ces deux enveloppes convexes, nous pourrions considérer les $(b' - b) + 1$ points entiers situés juste au-dessus de la droite L et les $(b' - b) + 1$ points entiers situés juste en dessous de la droite L, à savoir les droites discrètes de L. Puis, nous pourrions utiliser un des algorithmes généraux de calcul d'enveloppes convexes existants (voir la section 1.3.2). Un des plus célèbres s'appelle l'algorithme du *gift wrapping* et sa complexité en temps est de l'ordre de $O((b' - b)h)$. Cependant, nous préfèrerions utiliser la méthode proposée par Chan en 1996 qui atteint une complexité en $O((b' - b) \log h)$ ce qui est optimal dans le cas général, i.e. en considérant un nuage de points (voir [Jar73] et [Cha96]). Cependant, comme les points entiers qui nous intéressent sont les points d'une grille régulière situés au-dessus et en dessous d'un segment de droite, nous pouvons appliquer des algorithmes beaucoup plus spécifiques pour calculer les enveloppes convexes. Nous commençons par introduire deux approches existantes, l'une utilise la théorie des nombres et l'autre met en oeuvre des techniques de géométrie algorithmique. Ces algorithmes atteignent respectivement des complexités en temps de $O(\log b')$ et de $O(\log(b' - b))$, ce qui est bien meilleur que la complexité des approches dites générales.

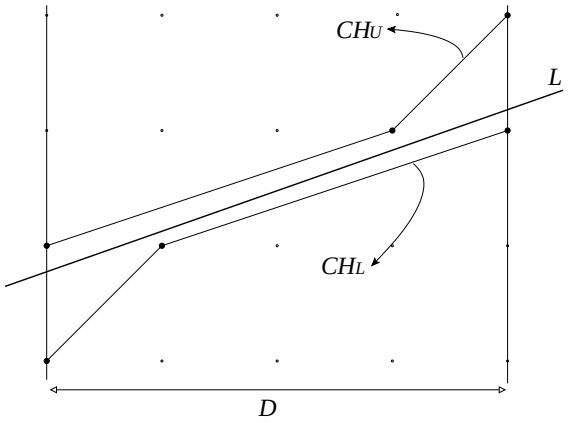


FIG. 2.1 – Les deux enveloppes convexes

2.3 Résoudre un Sous-Problème avec les Fractions Continues

Dans cette section, le sous-problème qui nous intéresse est le suivant : déterminer le nombre rationnel p/q qui approxime au mieux le nombre réel a et tel que q n'excède pas la valeur entière b'. Nous montrons comment résoudre ce problème numériquement et géométriquement en utilisant les fractions continues.

Nous remarquons que le sous-problème de cette section peut facilement être résolu en calculant les deux suites S_P et S_I issues du développement en fraction continue de a (voir section 1.2.2). D'après la proposition 2, nous savons que le dernier terme de la suite S_P (resp. S_I) dont le dénominateur ne dépasse pas b' correspond à la meilleure approximation rationnelle de a inférieure (resp. supérieure) à a. Nous illustrons cette proposition avec l'exemple 2).

Exemple 2 *Nous souhaitons déterminer le nombre rationnel dont le dénominateur n'excéde pas* 80 *et qui approxime au mieux le rationnel* 779/207. *Nous avons le développement en fraction continue suivant :* $779/207 = [3, 1, 3, 4, 2, 5]$. *Nous pouvons ainsi calculer les deux suites :*

$S_P = \frac{0}{1}, \frac{1}{1}, \frac{2}{1}, \frac{3}{1}, \frac{7}{2}, \frac{11}{3}, \frac{15}{4}, \frac{79}{21}, \frac{143}{38}, \frac{779}{207}$

$S_I = \frac{1}{0}, \frac{4}{1}, \frac{19}{5}, \frac{34}{9}, \frac{49}{13}, \frac{64}{17}, \frac{207}{55}, \frac{350}{93}, \frac{493}{131}, \frac{636}{169}, \frac{779}{207}$

De ces suites, nous déduisons que le rationnel 143/38 *de la suite* S_P *(resp. le rationnel* 207/55 *de la suite* S_I*) correspond à la meilleure approximation de* 779/207 *inférieure (resp. supérieure) à* 779/207, *dont le dénominateur ne dépasse pas* 80. *À partir de ces deux fractions, nous en déduisons que le nombre rationnel* 143/38 *correspond à la meilleure approximation de* 779/207 *telle que son dénominateur ne dépasse pas* 80.

Dans le plan euclidien, ce sous-problème est équivalent à déterminer le point entier (q, p) qui approxime au mieux la droite d'équation $y = ax$ tel que son abscisse n'excède pas b'. Nous pouvons interpréter géométriquement la fraction continue associée à un nombre réel a en faisant correspondre les nombres rationnels avec des points entiers

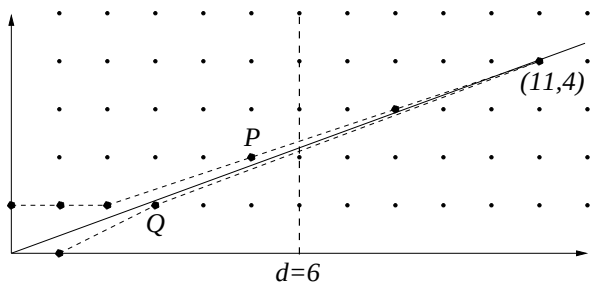


FIG. 2.2 – Exemple de voiles de Klein pour $a = 4/11$ et $d = 6$

et ainsi interpréter la Proposition 2 comme la Proposition 3 dans le plan euclidien (voir la section 1.2.2). Le meilleur point approximant que nous cherchons correspond donc au dernier terme de la série K_L ou de la série K_U dont l'abscisse n'excède pas b'. La figure 2.2 montre un exemple de suites de points entiers K_L et K_U pour $a = 4/11$. Dans cet exemple, les points entiers $P = (5, 2)$ et $Q = (3, 1)$ correspondent aux points approximant au mieux la droite de pente $4/11$ tels que leur abscisse ne dépasse pas 6.

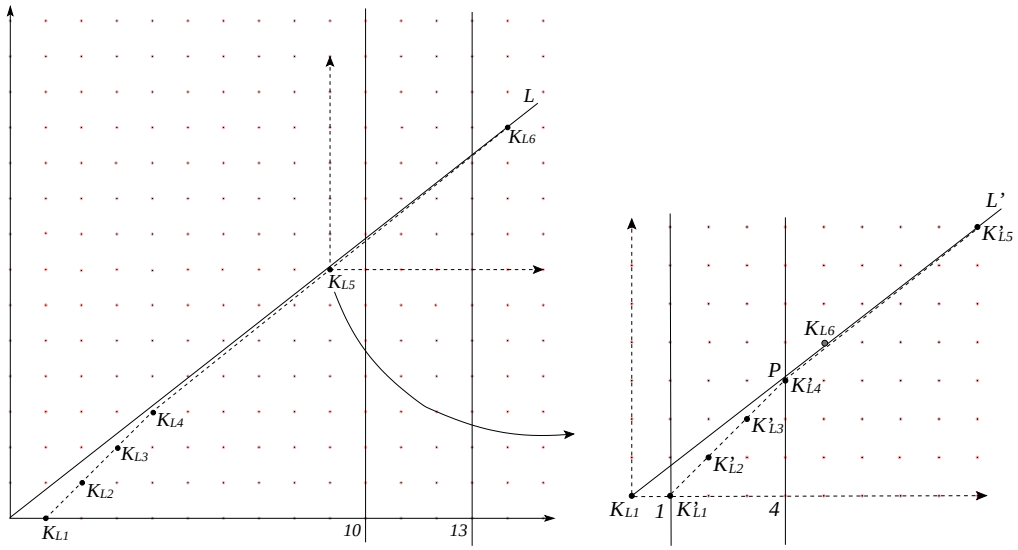
2.4 Résoudre le Problème d'Approximation avec les Fractions Continues

À partir de maintenant, nous nous intéressons à un problème plus contraint. Nous souhaitons déterminer le nombre rationnel p/q qui approxime au mieux le nombre réel a et tel que son dénominateur q appartient à l'intervalle borné $[b, b']$. En réalité, nous traitons la version géométrique de ce problème : déterminer le point entier (q, p) du domaine $D = \{(x, y) \in \mathbb{R}^2 | b \leq x \leq b'\}$ qui approxime au mieux la droite L d'équation $y = ax$.

2.4.1 Description de l'Algorithme

Pour résoudre ce problème, nous pouvons toujours utiliser les fractions continues et l'interprétation géométrique de la proposition 2 à la manière d'Harvey dans [Har99]. Pour cela, nous calculons la voile de Klein supérieure et inférieure de la droite L. Puis, nous sélectionnons le point entier de plus grande abscisse appartenant au domaine D sur chaque voile de Klein et nous les comparons. Malheureusement, il est possible qu'aucun point entier porté par les voiles de Klein ne se situe dans le domaine D. Soit t l'indice du point entier de la suite K_L ou de la suite K_U d'abscisse maximale mais toutefois inférieure à b'. Supposons qu'il s'agit d'un point de K_L et notons ce point K_{L_t}. Si le point K_{L_t} ou un point de la forme αK_{L_t} avec $\alpha \in \mathbb{N}$ appartient au domaine D, d'après la proposition 3, il correspond forcément au meilleur point approximant que nous recherchons et l'algorithme se termine. Dans le cas contraire, l'algorithme réitère. Pour cela, nous translatons l'origine du repère au point entier βK_{L_t} où β est une valeur entière positive telle que βK_{L_t} corresponde au point le plus proche de la droite d'équation $x = b$, à gauche du domaine D. Nous réitérons avec la droite L' de vecteur directeur K_{L_t} et avec le domaine D' de la forme $\{(x, y) \in \mathbb{R}^2 | b - \beta x_{L_t} \leq x \leq b' - \beta x_{L_t}\}$ où x_{L_t} correspond à l'abscisse du point K_{L_t}.

Nous pouvons remarquer que, lorque nous réitérons, aucun point entier du domaine D ne se situe entre les droites L et L' (voir Proposition 3). Par conséquent, nous ne manquons aucun point entier susceptible de correspondre au meilleur point approximant



(a) Première itération (b) Seconde itération

FIG. 2.3 – Exemple de déroulement de l'algorithme de Harvey

recherché. De plus, le calcul de la voile de Klein lors des itérations suivantes se fait en temps constant. En effet, les points entiers portés par cette voile correspondent exactement au t-ième premiers termes de la voile de Klein à l'itération précédente.

2.4.2 Exemple de Déroulement et Discussion sur l'Algorithme

Un court exemple du déroulement de l'algorithme est illustré par la figure 2.3. Dans cet exemple, nous considérons la droite L d'équation $y = (11/14)x$ ainsi que le domaine D de la forme $\{(x, y) \in \mathbb{R}^2 | 10 \leq x \leq 13\}$. La voile de Klein inférieure de L correspond à la suite de points entiers $[(1, 0), (2, 1), (3, 2), (4, 3), (9, 7), (14, 11)]$ et elle est illustrée en figure 2.3.a. Comme aucun terme de cette série ne correspond à une solution du problème d'approximation, nous réitérons l'algorithme. Nous translatons l'origine du repère au point $K_{L_5} = (9, 7)$. Dans ce nouveau repère, nous considérons le nouveau domaine D' défini par $\{(x, y) \in \mathbb{R}^2 | 1 \leq x \leq 4\}$ ainsi que la droite L' de vecteur directeur $(9, 7)$ comme sur la figure 2.3.b. La voile de Klein inférieure associée à la nouvelle droite L' correspond à la suite de points $[(1, 0), (2, 1), (3, 2), (4, 3), (9, 7)]$. Les quatre premiers points de cette suite appartiennent au domaine D', donc le point $P = K_{L'_4} = (4, 3)$ est solution de notre problème d'approximation. Par conséquent, dans le repère initial, le meilleur point approximant la droite L de pente $11/14$ d'abcisse à la fois supérieure à 10 et inférieure à 13 correspond au point $(13, 10)$.

La méthode de Harvey nécessite la connaissance d'un point à coordonnées entières porté par la droite à approximer, point considéré comme origine du repère. Son algorithme doit alors calculer dans le pire cas $O(\log(b'))$ convergents de la pente de la droite L. Notons que la théorie des nombres et en particulier les fractions continues peuvent probablement résoudre le problème d'approximation pour des droites ne passant pas par l'origine [AB98]. En effet, en utilisant le système de numération d'Ostrowski [Ost22] à la manière de Berthé et Imbert dans [BI09], nous voyons que l'intercept de la droite L (c'est-à-dire l'ordonnée de son intersection avec l'axe des y) peut être exprimée en fonction du développement en fractions continues de la pente de la droite L. Il existe ainsi peut être un moyen

d'approximer des droites ne passant pas par l'origine du repère en utilisant des outils de la théorie des nombres.

Nous remarquons que la valeur b' peut être beaucoup plus grande que la valeur $(b' - b)$ qui correspond à la largeur du domaine vertical. Dans ce cas, nous devons améliorer la complexité en temps pour le calcul de la meilleure approximation. En effet, nous souhaitons résoudre ce problème en $O(\log(b' - b))$ dans le pire cas. Bien que des outils de la théorie des nombres puisse peut être mener à cette complexité optimale, ils peuvent s'avérer assez complexes à utiliser dans ce cas. Nous introduisons donc une seconde approche utilisant la géométrie algorithmique.

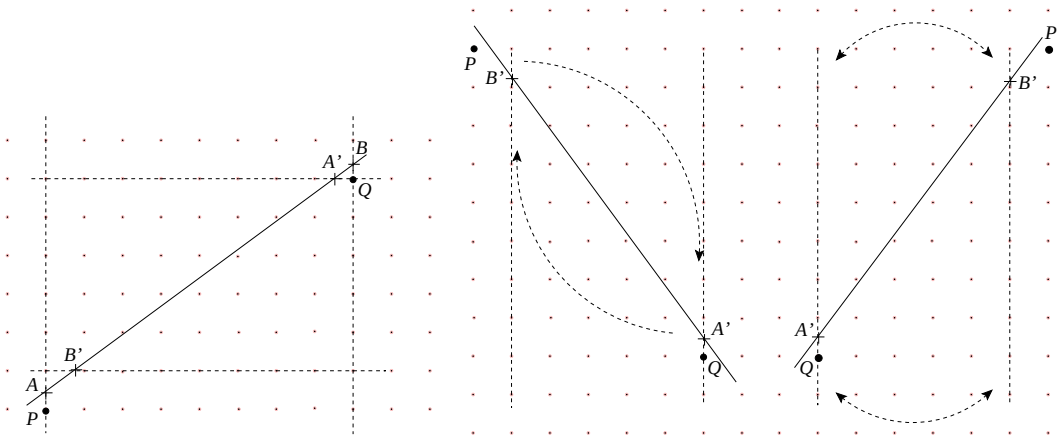
2.5 Une Seconde Approche Utilisant la Géométrie Algorithmique

Les notations $\lfloor x \rfloor$ et $\lceil x \rceil$ représentent respectivement les valeurs entières inférieures et supérieures du nombre réel x. Nous rappelons que la valeur entière inférieure de x correspond au plus grand nombre entier plus petit que x. De la même manière, la valeur entière supérieure de x correspond au plus petit nombre entier supérieur à x.

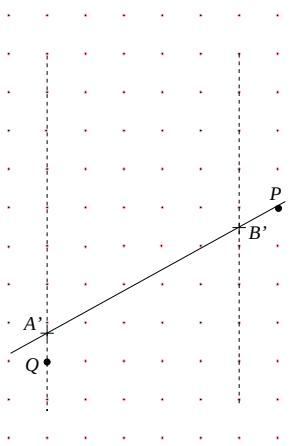
En 1999, Balza-Gomez et al. ont proposé dans [BGMM99] une méthode pour calculer l'enveloppe convexe CH_U ou CH_L décrite dans la section 2.2. Nous présentons le calcul de CH_L. Leur méthode atteint une complexité en temps de $O(\log(b' - b))$ et fonctionne en deux étapes. Notons AB le segment de droite correspondant à l'intersection de la droite L avec le domaine vertical D de sorte que A se trouve à gauche de B. La première étape consiste à calculer une pré-enveloppe associée au segment de droite AB qui comprend les sommets de l'enveloppe convexe finale mais également des points entiers supplémentaires. La seconde étape consiste à parcourir cette pré-enveloppe afin d'en supprimer les points supplémentaires. Notons que cette méthode ne nécessite pas que la droite L passe par l'origine du repère.

Nous décrivons tout d'abord le calcul de la pré-enveloppe associée au segment de droite AB. Cette pré-enveloppe commence au point entier situé juste en dessous de A (ou le point A lui-même) et se termine par le point entier situé juste en dessous de B (ou le point B lui-même). Nous calculons itérativement cette pré-enveloppe. À chaque itération, l'algorithme trouve deux points entiers et les concatène à l'enveloppe courante. Ensuite, nous déterminons un segment de droite plus petit $A'B'$ porté par AB et correspondant au segment de droite considéré à l'itération suivante. L'algorithme se termine lorsque la taille du segment de droite courant est réduite à zéro.

À chaque itération, les deux points ajoutés à la pré-enveloppe sont de la forme $(x_A, \lfloor y_A \rfloor)$ et $(x_B, \lfloor y_B \rfloor)$. Ensuite, pour déterminer le segment de droite considéré à l'itération suivante, nous procédons comme suit. Nous considérons les deux droites horizontales d'équation $y = \lceil y_A \rceil$ et $y = \lfloor y_B \rfloor$ ainsi que leurs intersections respectives B' et A' avec le segment de droite AB. Nous appliquons alors une rotation d'angle $-\pi/2$ radians puis une symétrie verticale. Comme la pente de la droite portée par A' et B' est maintenant strictement supérieure à 1, nous appliquons une transformation unimodulaire $U : (x, y) \rightarrow (x', y')$ de $\mathbb{Z}^2$ de façon à ce que la valeur absolue de la pente de la droite portée par les transformées de A' et de B' soit inférieure à 1 (voir la section 1.1.2). Les équations définissant cette



(a) Sélection du sous-segment $A'B'$ (b) Après la rotation (c) Après la symétrie



(d) Après l'étape de réduction

FIG. 2.4 – Exemple d'une itération

transformation et la matrice correspondante sont de la forme :

$$\begin{cases} x' = x \\ y' = y - \lfloor a' \rfloor x \end{cases} \quad \Leftrightarrow \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\lfloor a' \rfloor & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

où a' représente la pente de la nouvelle droite passant par A' et B'. Cette transformation correspond à l'étape dite de réduction car elle diminue la longueur du segment $A'B'$ à chaque itération.

La figure 2.4 illustre un exemple d'une itération de la première étape de l'algorithme. Nous commençons par ajouter à la pré-enveloppe les deux points entiers P et Q et nous déterminons le segment de droite $A'B'$ (figure 2.4.a). Ensuite, nous appliquons une rotation d'angle $-\pi/2$ radians (figure 2.4.b) et la symétrie verticale (figure 2.4.c). Enfin, nous appliquons l'étape de réduction afin de pouvoir réitérer avec le segment de droite $A'B'$ (figure 2.4.d).

Afin de supprimer les points supplémentaires ajoutés à la pré-enveloppe pendant la première étape, nous exécutons la seconde phase de l'algorithme. Nous appliquons la méthode du *Graham scan* (voir [Gra72] et la section 1.3.2) sur la pré-enveloppe afin de construire l'enveloppe convexe entière finale.

Les auteurs ont prouvé que la première étape effectue $O(\log(b'-b))$ itérations, chaque itération s'exécutant en temps constant et ajoutant au plus deux points entiers à la pré-enveloppe. De plus, comme les points entiers de la pré-enveloppe sont rangés par abscisses croissantes, la seconde étape du Graham scan est suffisante pour calculer l'enveloppe convexe finale en temps linéaire suivant le nombre de points de la pré-enveloppe. La méthode s'exécute donc en $O(\log(b'-b))$ mais sa complexité ne dépend pas de la taille de la sortie. En effet, le nombre de points supplémentaires ajoutés à la pré-enveloppe, et donc le nombre d'itérations ne peut pas être exprimé en fonction du nombre de sommets de l'enveloppe convexe finale. Le nombre d'itérations de cet algorithme n'est donc pas linéaire en le nombre de sommets de l'enveloppe convexe finale.

2.6 Notre Nouvelle Méthode pour le Calcul des Enveloppes Convexes

La méthode décrite dans la section 2.4 utilisant la théorie des nombres atteint une complexité en $O(\log(b'))$ tandis que la méthode décrite dans la section 2.5 mettant en oeuvre des techniques de géométrie algorithmique s'exécute en $O(\log(b'-b))$, soit la complexité optimale. Nous proposons une nouvelle approche pour calculer les enveloppes convexes CH_U et CH_L qui combine à la fois la théorie des nombres et la géométrie algorithmique. Notre méthode est la première qui préserve la complexité optimale et dont le nombre d'itérations peut s'exprimer en fonction de la taille de la sortie, notre algorithme est donc adaptatif. En effet, notre algorithme calcule simultanément les deux enveloppes convexes en temps linéaire suivant le nombre de leurs sommets. En outre, comme l'approche précédente, notre méthode ne nécessite pas que la droite L passe par l'origine du repère, il suffit de connaître son équation et la forme du domaine vertical.

2.6.1 Vue d'Ensemble de Notre Algorithme

L'algorithme que nous proposons calcule simultanément les deux enveloppes convexes CH_U et CH_L par rapport à la droite L et au domaine borné D. Chaque enveloppe convexe peut être décomposée en deux parties : une partie gauche et une partie droite. Soit N le vecteur normal de la droite L orienté vers le haut, c'est-à-dire dans la direction de CH_U. La partie gauche de CH_U s'étend de son sommet le plus à gauche à son sommet extrémal dans la direction $-N$, de plus petite abscisse. De manière analogue, la partie droite de CH_U s'étend de son sommet le plus à droite à son sommet extrémal dans la direction $-N$, de plus grande abscisse. Notons que la partie droite et la partie gauche peuvent avoir un sommet en commum. Nous pouvons décrire de façon similaire la partie gauche et la partie droite de CH_L. Soient L_{left} et L_{right} les droites verticales respectivement d'équation $x = b$ et $x = b'$, bornant le domaine vertical D. La figure 2.5 illustre les différentes parties des enveloppes convexes.

Notre algorithme calcule simultanément les sommets des deux enveloppes convexes CH_U et CH_L et procède en deux phases. La première phase consiste à déterminer itérativement les sommets des parties gauches de CH_U et de CH_L depuis L_{left} vers la droite. La seconde phase consiste à calculer les sommets des parties droites de CH_U et CH_L depuis L_{right} vers la gauche, en utilisant le même algorithme. Chaque sommet déterminé durant la première ou durant la seconde phase est plus proche de la droite de référence L que

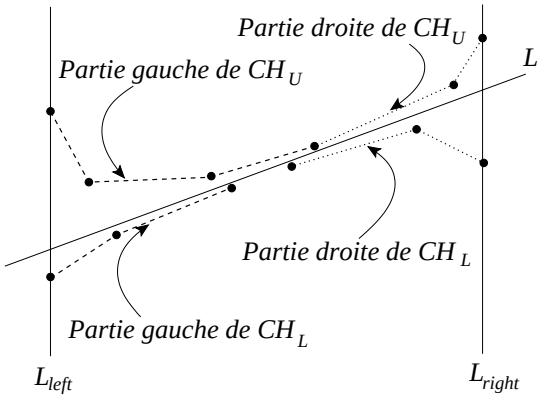


FIG. 2.5 – Séparation des enveloppes convexes en parties

le sommet précédent. Par conséquent, le meilleur point approximant que nous cherchons correspond à l'un des points extrémaux des enveloppes convexes.

Notons respectivement CH_U^L et CH_L^L les parties gauches des enveloppes CH_U et CH_L. Nous décrivons la première étape de notre méthode à savoir le calcul de CH_U^L et CH_L^L. La seconde étape est similaire à la première. Notons A_1 et B_1 les deux premiers sommets de CH_U^L et de CH_L^L respectivement. Ces sommets correspondent aux deux points entiers portés par la droite verticale L_{left} situés respectivement juste au-dessus et juste en dessous de l'intersection de la droite de référence L avec L_{left}. Supposons que nous avons trouvé le i-ème sommet A_i de CH_U et le j-ème sommet B_j de CH_L. Notre méthode détermine en temps constant le sommet A_{i+1} de CH_U ou le sommet B_{j+1} de CH_L. Pour cela, nous cherchons un point entier K appartenant au domaine D tel que la valeur absolue de l'angle entre $A_{i-1}A_i$ et A_iK lorsque K est situé au-dessus de L ou entre $B_{j-1}B_j$ et B_jK lorsque K est situé en dessous L est minimal. Dans la section suivante, nous détaillons la manière dont nous déterminons ce sommet.

2.6.2 Déroulement de la Méthode

Nous déterminons le sommet suivant A_{i+1} ou B_{j+1} en utilisant les sommets déjà calculés A_i et B_j ainsi que deux vecteurs particuliers notés u_{ij} et v_{ij}. À chaque itération, le vecteur u_{ij} correspond au vecteur A_iB_j et le vecteur v_{ij} correspond au vecteur de Bezout de u_{ij} tel que le point entier $A_i + v_{ij}$ soit situé au-dessus de L et tel que le point entier $B_j + v_{ij}$ soit situé en dessous de L (voir la figure 2.6.a ou 2.6.b). Nous appelons ce vecteur v_{ij} le *vecteur de Bezout valide* de u_{ij} pour la droite L. Notre méthode est présentée de manière simplifiée dans l'algorithme 9. En effet, nous ne montrons que les étapes principales et nous considérons que la droite L n'est pas portée par des points entiers du domaine D. La fonction *Intersection* de paramètres (P, w, L) calcule le point d'intersection de la droite passant par le point P et de vecteur directeur w avec la droite L. Ce point d'intersection est de la forme $P + \alpha w$ avec α un nombre réel. La fonction *Intersection* renvoie la valeur entière $\lfloor \alpha \rfloor$.

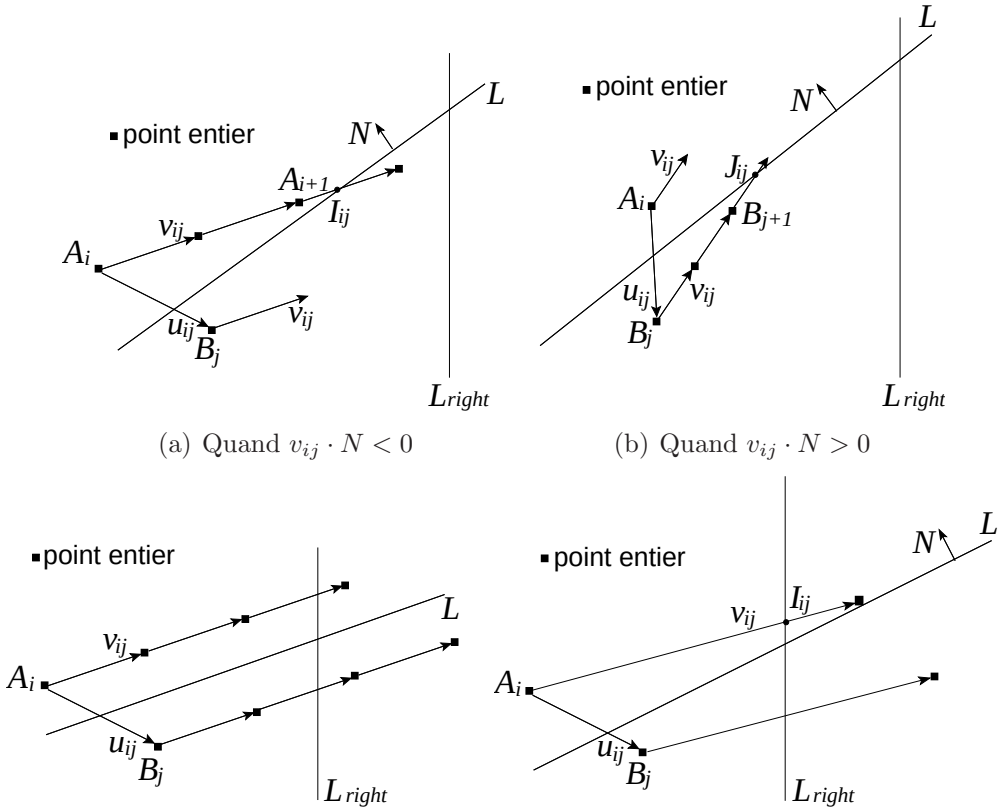
À chaque itération, nous distinguons trois configurations différentes dépendant du signe du produit scalaire $v_{ij} \cdot N$. Nous détaillons à présent ces trois configurations possibles. La première configuration se produit lorsque le produit scalaire $v_{ij} \cdot N$ est strictement négatif. Considérons les points d'intersection de la droite de vecteur directeur v_{ij} passant par le point A_i avec la droite de référence L et avec la droite verticale L_{right}. Notons I_{ij} le

Algorithme 9 : Notre algorithme de calcul d'enveloppe convexe : Première étape

```
1 Init(A_1,B_1,u_11,v_11)
2 i ← 1 j ← 1
3 CONTINUE←VRAI
4   TANT QUE (CONTINUE)
5     SI(v_ij · N < 0)
6       ⌊β⌋ ← min(Intersection(A_i, v_ij, L),Intersection(A_i, v_ij, L_right))
7       SI(⌊β⌋ ≥ 1)
8         A_{i+1} ← A_i + ⌊β⌋v_ij
9         i ← i + 1
10        MettreAJour(u_ij, v_ij)
11      SINON
12        CONTINUE ← FAUX
13    SI(v_ij · N > 0)
14      ⌊β⌋ ← min(Intersection(B_j, v_ij, L),Intersection(B_j, v_ij, L_right))
15      SI(⌊β⌋ ≥ 1)
16        B_{j+1} ← B_j + ⌊β⌋v_ij
17        j ← j + 1
18        MettreAJour(u_ij, v_ij)
19      SINON
20        CONTINUE ← FAUX
21    SINON
22      CONTINUE ← FAUX
```

point le plus à gauche parmi ces deux points d'intersection. Le point I_{ij} est de la forme $A_i + \alpha v_{ij}$ avec $\alpha \in \mathbb{R}^+$. Par définition des vecteurs de Bezout, le parallélogramme $A_i, A_i + \alpha v_{ij}, B_j + \alpha v_{ij}, B_j$ ne contient aucun point entier. Puisque le triangle $A_i I_{ij} B_j$ est inclus dans ce parallélogramme, lui aussi ne contient aucun point entier. Par conséquent, si α est supérieur à 1, le point entier $A_i + \lfloor \alpha \rfloor v_{ij}$ correspond au nouveau sommet A_{i+1} de CH_U^L (voir figure 2.6.a). Si α est strictement plus petit que 1, alors A_i correspond au dernier sommet de CH_U^L et la première étape se termine (voir figure 2.6.d). La seconde configuration se produit lorsque le produit scalaire $v_{ij} \cdot N$ est strictement positif. Considérons les points d'intersection de la droite de vecteur directeur v_{ij} passant par le point B_j avec la droite de référence L et avec la droite verticale L_{right}. Notons J_{ij} le point le plus à gauche parmi ces deux points d'intersection. Le point J_{ij} est de la forme $B_j + \beta v_{ij}$ avec $\beta \in \mathbb{R}^+$. Pour les mêmes raisons que précédemment, si β est supérieur à 1, le point entier $B_j + \lfloor \beta \rfloor v_{ij}$ correspond au nouveau sommet B_{j+1} de CH_L^L (voir figure 2.6.b). Si β est strictement plus petit que 1, alors B_j correspond au dernier sommet de CH_L^L et la première étape se termine (voir figure 2.6.d). Une troisième configuration peut également se produire, si le produit scalaire $v_{ij} \cdot N$ vaut zéro. Dans ce cas, le dernier sommet calculé de CH_U^L et le dernier sommet calculé de CH_L^L correspondent en fait aux derniers sommets des parties gauches des enveloppes convexes et la première étape se termine donc (voir figure 2.6.c).

Soient respectivement P_U^L et P_L^L les sommets de CH_U et de CH_L extrémaux dans la direction $-N$ et dans la direction N, de plus petite abscisse. Ils correspondent respectivement aux derniers sommets de CH_U^L et de CH_L^L. Nous prouvons que la première étape de notre algorithme se termine lorsque le dernier sommet calculé de CH_U^L et le dernier sommet calculé de CH_L^L correspondent respectivement aux deux points extrémaux P_U^L et P_L^L.



(a) Quand $v_{ij} \cdot N < 0$

(b) Quand $v_{ij} \cdot N > 0$

(c) Quand $v_{ij} \cdot N = 0$

(d) Exemple de fin de l'algorithme en première configuration

FIG. 2.6 – Les trois configurations de notre méthode et un exemple de fin de l'algorithme

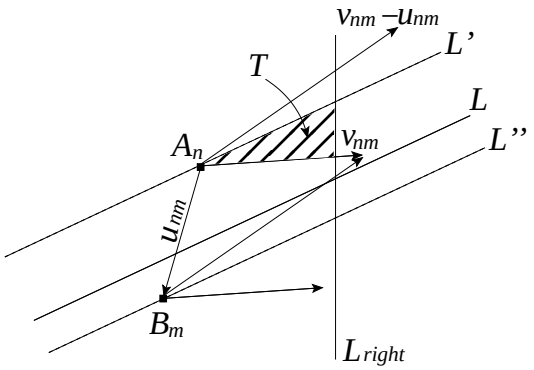


FIG. 2.7 – Exemple de terminaison de l'algorithme

Notons respectivement A_n et B_m les derniers sommets calculés de CH_U et de CH_L, à la fin de la première étape. Notons L' et L'' les droites parallèles à L et passant respectivement par les points A_n et B_m. Afin de prouver que les points A_n et B_m correspondent bien aux points extrémaux P_U^L et P_L^L, nous montrons qu'aucun point entier du domaine D ne se situe strictement entre les droites L' et L''. Supposons sans perte de généralité que lorsque la première étape se termine, le produit scalaire $v_{nm} \cdot N$ est strictement négatif comme sur la figure 2.7. Ceci implique que le segment de droite $A_n\ A_n + v_{nm}$ intersecte la droite L_{right}. Nous savons que les suites de points $A_1, A_2, \cdots, A_n$ et $B_1, B_2, \cdots, B_m$ représentent respectivement une partie des sommets de CH_U et de CH_L. Ceci implique qu'aucun point entier du domaine D, situés à gauche de A_nB_m, n'est compris entre les droites L' et L''. De plus, par définition des vecteurs de Bezout, nous savons qu'aucun point entier ne se situe dans le parallélogramme de la forme $(A_n, A_n + \gamma v_{nm}, B_m + \gamma v_{nm}, B_m)$ quel que soit $\gamma \in \mathbb{N}$. Par conséquent, soit T le triangle borné par L', L_{right} et la droite passant par A_n de vecteur directeur v_{nm}, il reste à prouver que T ne contient aucun point entier. Pour cela, nous remarquons tout d'abord que la droite L intersecte les segments de droite ouverts (A_n, B_m) et (A_n+v_{nm}, B_m+v_{nm}). De plus, comme le produit scalaire $v_{nm}\cdot N$ est strictement négatif, ceci implique que la droite L' passant par B_m intersecte également le segment de droite ouvert $(A_n + v_{nm}, B_m + v_{nm})$. Par translation, nous en déduisons que la droite L' passant par A_n intersecte le segment de droite ouvert $(A_n + v_{nm} - u_{nm}, B_m + v_{nm} - u_{nm})$. Par conséquent, le triangle T est entièrement contenu dans une bande définie par les deux droites de direction $v_{nm} - u_{nm}$ et passant respectivement par A_n et par B_m. Comme cette bande ne contient aucun point entier, il en est de même pour le triangle T. Nous pouvons prouver de la même manière que la seconde étape se termine lorsque tous les sommets des parties droites de CH_U et de CH_L ont été déterminés.

2.6.3 Analyse de Complexité

Dans cette section, nous décrivons la complexité en temps de la première étape de l'algorithme, i.e. le calcul de CH_U^L et CH_L^L. La seconde étape est analogue. Les deux premiers sommets A_1 et A_2 correspondent aux points entiers portés par L_{left} situés juste au-dessus et juste en dessous de L respectivement. Pour les calculer, il suffit de déterminer l'intersection de la droite verticale L_{left} avec la droite L en temps constant.

À chaque itération, pour déterminer un nouveau sommet de CH_U^L ou de CH_L^L, nous commençons par mettre à jour les deux vecteurs u_{ij} et v_{ij}. Puis, nous calculons un nouveau sommet de CH_U^L ou de CH_L^L en utilisant la fonction *Intersection*, soit avec la droite définie

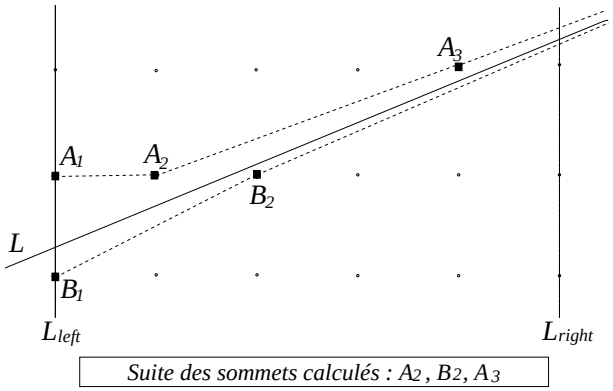


FIG. 2.8 – La première étape de l'algorithme

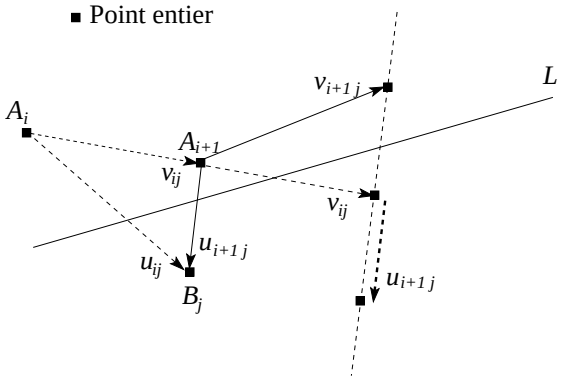


FIG. 2.9 – La mise à jour de $v_{i+1\ j}$

par A_i et v_{ij} soit avec la droite définie par B_j et v_{ij}, et avec la droite L. La fonction *Intersection* s'exécute en temps constant et nous calculons donc chaque nouveau sommet en $O(1)$. Étant donné que chaque enveloppe convexe comprend au plus $\lceil 1+\log((b'-b)+1)\rceil$ sommets (voir [Zol00]), notre algorithme traite $\log(b'-b)+O(1)$ iterations et s'exécute donc en $O(\log(b'-b))$. De plus, notre méthode est dite adaptative car elle calcule un nouveau sommet à chaque itération en temps constant. La complexité de notre algorithme est donc linéaire suivant le nombre de sommets des enveloppes convexes. La figure 2.8 montre un exemple d'exécution de la première étape de notre méthode.

Remarque 3 *À chaque itération, nous faisons une mise à jour des vecteurs u_{ij} et v_{ij} en temps constant. En effet, supposons sans perte de généralité que nous avons déterminé le sommet A_{i+1} de CH_U^L. Nous commençons l'itération suivante et le vecteur $u_{i+1\ j}$ correspond au vecteur $A_{i+1}B_j$ de la forme $u_{ij}+\gamma v_{ij}$, où γ représente une valeur entière. Nous devons alors déterminer le vecteur de Bezout valide $v_{i+1\ j}$ de $u_{i+1\ j}$. Par définition des vecteurs de Bezout (voir la section 1.2.1), en connaissant un vecteur de Bezout particulier de $u_{i+1\ j}$, nous pouvons en déduire tous ses autres vecteurs de Bezout. Cependant, le vecteur v_{ij} correspond toujours à un vecteur de Bezout du vecteur $u_{i+1\ j}$ car $(u_{ij}+\gamma v_{ij})\wedge v_{ij} = u_{ij}\wedge v_{ij} = 1$. Par conséquent, nous pouvons déterminer le vecteur de Bezout valide $v_{i+1\ j}$ de $u_{i+1\ j}$ en temps constant en appelant la fonction Intersection avec les paramètres $A_{i+1}+v_{ij}$, $u_{i+1\ j}$ et L. Soit $\lfloor\alpha\rfloor$ la valeur retournée par cette fonction Intersection, nous déduisons que $v_{i+1\ j}$ correspond au vecteur $v_{ij}+\lfloor\alpha\rfloor u_{i+1\ j}$ (voir la figure 2.9). Cette étape de mise à jour s'exécute donc en temps constant.*

2.7 Exemples d'Application

Notre méthode décrite dans la section 2.6 pour calculer les deux enveloppes convexes CH_U et CH_L fonctionne également lorsque la droite L ne passe pas par l'origine. Nous pouvons donc mettre en évidence des applications générales de notre algorithme, par exemple dans le domaine de la compression de données ou de la programmation linéaire en nombres entiers.

2.7.1 Application à la Compression de Données

Considérons une droite L d'équation $\alpha x + \beta y = \gamma$ à coefficients irrationnels, ou plus généralement à coefficients réels de grande précision. Si nous restreignons notre espace de travail à un domaine rectangulaire borné de la forme $\{(x, y)|b \leq x \leq b' et c \leq y \leq c')\}$, par exemple l'écran de l'ordinateur, nous n'avons pas besoin de manipuler des coefficients beaucoup plus grands que la taille de l'écran. Nous introduisons les deux définitions suivantes :

Définition 9 *La droite à coefficients réels L' d'équation $\alpha' x + \beta' y = \gamma'$ est* équivalente *à la droite L d'équation $\alpha x + \beta y = \gamma$* par rapport *au domaine D si et seulement si tout point entier (x, y) du domaine D vérifie :*

$$\alpha x + \beta y \leq \gamma \Leftrightarrow \alpha' x + \beta' y \leq \gamma'$$

Définition 10 *La droite L' correspond à une* réduction *de la droite L par rapport au domaine D si et seulement si L' est équivalente à L par rapport à D et* $\max\{|\alpha'|, |\beta'|\}$ *ne dépasse pas la valeur $(b' - b)$ avec α' et $\beta' \in \mathbb{Z}$.*

Supposons sans perte de généralité que la valeur absolue de la pente de L est inférieure à 1. Ce problème de compression de données revient à réduire la droite L à une droite équivalente L' passant par au moins deux points entiers du domaine D de la forme $\{(x, y)|b \leq x \leq b'\}$. La transformation de la droite peut donc être effectuée en calculant d'abord l'enveloppe convexe inférieure et supérieure des points de la grille par rapport à la droite L et au domaine D comme décrit dans la section 2.6. Ensuite, la droite réduite correspond à une des deux *droites supports critiques* de ces deux enveloppes convexes (voir figure 2.10).

Définition 11 *Les* droites supports critiques *de deux polygones convexes correspondent aux deux droites tangentes aux deux polygones, telles que les polygones se situent de part et d'autre des droites.*

D'après [Tou83], nous calculons une de ces deux droites supports critiques en temps linéaire en le nombre de sommets des deux enveloppes convexes. Comme chaque enveloppe convexe compte $O(\log(b' - b))$ sommets (voir [Zol00]), ce calcul se fait en $O(\log(b' - b))$.

Exemple 3 *Considérons par exemple que nous travaillons sur un écran* 800×600 *avec un repère dont l'origine est située au coin inférieur gauche de l'écran. Considérons à présent une droite L d'équation $(\sqrt{2} - 1)x - y = -(5\sqrt{5})/2$. En calculant les deux enveloppes convexes entières de la droite L par rapport à l'écran, nous obtenons : CH_U = [(0,6), (3,7), (13,11), (71,35), (479,204), (718,303), (788,332), (800,337)] et CH_L = [(0,5), (1,6), (240,105), (648,274), (747,315), (776,327), (793,334), (798,336), (800,336)]. Une des droites critiques supports de ces deux enveloppes convexes passe par le sommet* $(718, 303)$ *de CH_U et par le sommet* $(240, 105)$ *de CH_L. Nous pouvons transformer la droite L en cette droite critique équivalente d'équation* $99x - 239y = -1335$.

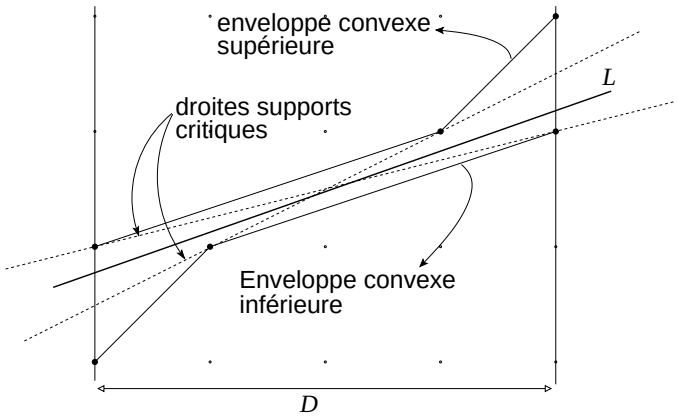


FIG. 2.10 – Droite réduite

2.7.2 Application à l'Optimisation Convexe

Soit $f : \mathbb{R}^d \to \mathbb{R}$ une fonction convexe. Résoudre un problème d'optimisation sur une telle fonction f revient à rechercher le point x tel que $f(x)$ soit minimum ou maximum. Supposons que nous nous intéressons aux solutions x à coordonnées entières du problème d'optimisation. Une méthode possible pour résoudre un tel problème consiste à partir d'un domaine de recherche correspondant à un polygone convexe à sommets entiers, puis à sélectionner à l'itération i un point x^i à l'intérieur de l'espace de recherche. Nous calculons ensuite un gradient ou un sous-gradient $\nabla(x^i)$ de la fonction f en ce point. Par définition du gradient d'une fonction convexe, tout point $x' \in \mathbb{R}^d$ satisfait $f(x') - f(x^i) \geq \nabla(x^i) \cdot (x' - x^i)$. Le gradient indique la direction de plus forte pente de la fonction f au point x^i.

Par conséquent, cela signifie que le domaine de recherche peut être réduit suivant le gradient au point x^i. Pour cela, en 2D, nous coupons le domaine de recherche suivant la droite D de vecteur normal $\nabla(x^i)$ passant par le point x^i. Le polygone résultat est un polygone convexe mais ses sommets correspondent à des points à coordonnées rationnelles dont le numérateur et le dénominateur ne cessent de croître au fil des itérations. Afin de manipuler un domaine de recherche à sommets entiers à chaque itération, il est nécessaire de reconstruire après chaque coupe le domaine de recherche comme un polygone à sommets entiers sans perdre de points potentiellement solutions.

Cette reconstruction peut être effectuée en adaptant légèrement notre algorithme de construction d'enveloppe convexe. Notre algorithme actuel calcule l'enveloppe convexe des points entiers d'un domaine vertical borné, situés au-dessus et en dessous d'une droite. Pour adapter l'algorithme, il suffit de ne plus considérer un domaine vertical mais un domaine borné par les côtés de l'espace de recherche intersecté par la droite de coupe. Soient C_1 et C_2 les droites portées par ces côtés. Seule l'initialisation des premiers points des enveloppes diffèrent. Dans l'algorithme de base, le domaine est borné par des droites verticales et nous sélectionnons donc en temps constant les points entiers de ces droites verticales situés juste au-dessus et juste en dessous de leur intersection avec la droite de coupe. Si le domaine est borné par des droites non-verticales, nous devons utiliser l'algorithme d'Euclide étendu à partir des coefficients de ces droites pour initialiser les enveloppes convexes (exemple en figure 2.11). Cette initialisation ne se fait donc plus en temps constant mais en temps logarithmique en fonction des coefficients de droites C_1 et C_2. Dans le chapitre 4, nous mettons en oeuvre cet algorithme de reconstruction d'enveloppes convexes entières lors de la résolution d'un problème de réalisabilité sur une fonction convexe bidimensionnelle.

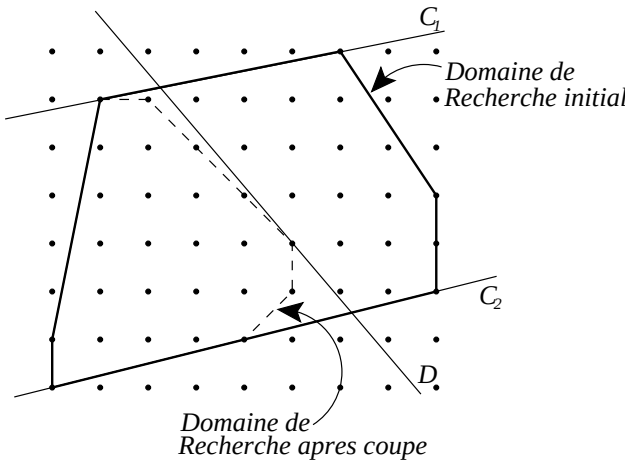


FIG. 2.11 – Reconstruction d'enveloppe convexe

2.8 Conclusion

La méthode exposée dans ce chapitre permet de déterminer de manière efficace le nombre rationnel approximant au mieux un nombre réel donné, tel que son dénominateur appartienne à un intervalle borné de la forme $[b, b']$. Dans le plan euclidien, ce problème est équivalent à déterminer le point entier (q, p) d'un domaine vertical borné D de la forme $\{(x, y) \in \mathbb{R}^2 | b \leq x \leq b'\}$ tel que la droite d'équation $y = (p/q)x$ passant par (q, p) approxime au mieux la droite L d'équation $y = ax$. Nous montrons que la détermination de ce point est liée avec le calcul de deux enveloppes convexes particulières. Nous décrivons deux approches existantes pour calculer ces enveloppes. La première utilise l'interprétation géométrique des fractions continues ([Har99]) et s'exécute en $O(\log(b'))$. Cependant, cette complexité n'est pas satisfaisante, surtout lorsque b' s'avère bien plus grand que la largeur du domaine vertical. La seconde approche met en oeuvre la géométrie algorithmique. L'algorithme proposé par Balza-Gomez et al. dans [BGMM99] pour calculer ces enveloppes s'exécute en $O(\log(b' - b))$ dans le cas général mais n'est pas adaptatif. Enfin, nous proposons notre méthode qui calcule en même temps l'enveloppe convexe inférieure et supérieure en $O(\log(b' - b))$. En outre, notre méthode calcule à chaque itération un nouveau sommet de l'enveloppe inférieure ou de l'enveloppe supérieure. Comme chaque itération s'exécute en temps constant, notre méthode est la seule à être également adaptative. En effet, sa complexité est linéaire suivant le nombre de sommets des enveloppes convexes finales. De plus, l'algorithme est simple à programmer et donc adapté pour l'implémentation. Notre méthode peut être vue comme une généralisation des voiles de Klein ne nécessitant pas que la droite à approximer passe par l'origine du repère. Dans la dernière section, nous mettons en évidence l'application de notre algorithme pour la compression de données. En effet, nous considérons une droite à coefficients irrationnels ou à coefficients réels de grande précision et nous montrons comment notre méthode peut être utilisée pour réduire les coefficients de cette droite par rapport à un domaine rectangulaire borné, comme l'écran de l'ordinateur. Notons que dans [BGMM99] Balza-Gomez et al. proposent une autre méthode pour calculer l'enveloppe convexe des points entiers appartenant à un domaine vertical entier borné et situés cette fois-ci en dessous d'une courbe convexe et bi-monotone. Leur algorithme est assez proche de notre méthode de calcul d'enveloppes convexes pour une droite et il atteint une complexité en $O(\log^2(b' - b))$ ou plus exactement une complexité adaptative en $O(k \log(b' - b))$ où k représente le nombre de sommets de

l'enveloppe convexe calculée. Nous pourrions facilement adapter notre algorithme de calcul d'enveloppes convexes au cas des courbes convexes comme Balza-Gomez et al. Dans cette configuration, notre algorithme fonctionne de la même manière que pour les droites mais la suite de points entiers $(A_i)_{1\leq i\leq m}$ situés au-dessus de la courbe et déterminés durant notre méthode ne correspondent plus à l'enveloppe convexe des points de la grille bordant supérieurement la droite. En effet, ils correspondent simplement aux sommets d'une ligne polygonale n'intersectant pas la courbe telle qu'aucun point de la grille ne se situe entre la courbe et la ligne polygonale. De plus, la complexité de notre méthode devient alors $O(k \log(b' - b))$ avec k le nombre de sommets de l'enveloppe convexe inférieure. En effet, pour déterminer chaque sommet de l'enveloppe convexe inférieure à la courbe il peut être nécessaire dans le pire cas de calculer $O(\log(b' - b))$ sommets de la ligne polygonale supérieure.

Chapitre 3

Introduction à la Reconnaissance de Plans Discrets

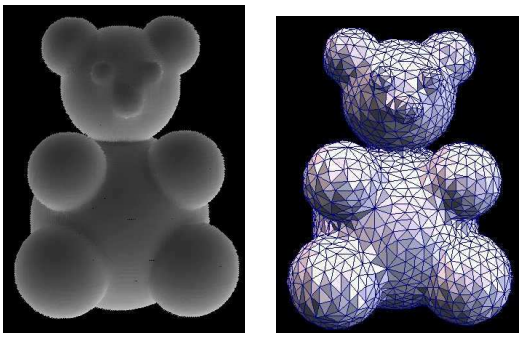
3.1 Introduction

À partir du moment où des images 2D ou 3D sont acquises par capteurs numériques (appareil photo numérique, scanner, IRM...), les objets réels se retrouvent alors échantillonnés sur des grilles entières régulières. De façon générale, les propriétés et théorèmes de la géométrie dans l'espace euclidien ne sont pas valables dans les espaces discrets. La géométrie discrète consiste à caractériser les objets discrets (courbure, surface, connexité...) et à mettre en place une algorithmique adaptée pour manipuler ces objets. Dans ce mémoire, nous nous intéressons en particulier à un domaine de recherche largement étudié : la reconnaissance de plans discrets (voir un récapitulatif dans [BCK07]). Cette problématique consiste à déterminer si un ensemble de points dans $\mathbb{Z}^3$, également appelé ensemble de voxels, correspond à la discrétisation d'un morceau de plan euclidien. La reconnaissance de plans discrets est donc utilisée pour décider si une partie d'un objet discret est plate, et par extension, pour polyédriser un objet discret. Les domaines applicatifs sont nombreux : imagerie médicale, réalité virtuelle, analyse de documents... En réalité virtuelle par exemple, un objet discret sous la forme d'un ensemble de voxels peut difficilement être manipulé tel quel car le volume de données est trop important. Généralement,nous transformons l'objet en un polyèdre, c'est-à-dire en une collection de facettes, et nous décidons de la précision de la polyédrisation en fonction de la profondeur de l'objet dans la scène virtuelle. En effet, un objet qui apparaît au loin dans une scène virtuelle pourra être remplacé par un polyèdre avec un nombre de facettes largement réduit par rapport à un objet positionné au premier plan. La figure 3.1 montre un ensemble de voxels et une polyédrisation possible de cet ensemble, les images sont tirées de [Bur02].

Nous introduisons tout d'abord des notations et définitions de bases dans la section 3.2 puis nous passons en revue quelques uns des algorithmes de reconnaissance de plans discrets dans la section 3.3.

3.2 Notations et Définitions de Base

Nous utilisons dans ce chapitre la notation $S = \{p^1, \ldots, p^n\}$ pour représenter un ensemble de n points de $\mathbb{Z}^d$. De plus, nous notons $(x_1, \ldots, x_d)$ les coordonnées d'un vecteur x dans $\mathbb{Z}^d$. Par la suite, comme nous traitons de reconnaissance de plans discrets, nous



(a) Ensemble de voxels (b) Polyèdre associé

FIG. 3.1 – Exemple de polyédrisation d'un objet discret

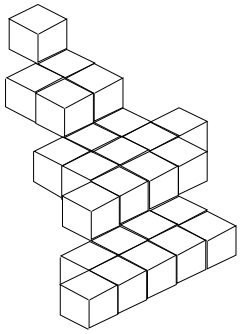


FIG. 3.2 – Ensemble de voxels

nous intéressons en particulier à des ensembles de points tridimensionnels. Notons qu'un objet discret 3D peut être représenté soit comme un ensemble de voxels comme sur la figure 3.2, soit comme un ensemble de points de $\mathbb{Z}^3$ généralement assimilés aux centres des voxels composant l'objet.

Un ensemble de points de $\mathbb{Z}^3$ est appelé morceau de plan discret s'il correspond à la discrétisation d'un morceau de plan euclidien. Comme un plan euclidien P est défini par son vecteur normal $N = (N_1, N_2, N_3)$, un plan discret correspondant à une discrétisation de P est également défini par le vecteur normal N. Nous remarquons qu'un morceau de plan discret correspond à la discrétisation d'une infinité de plans euclidiens résultant de petites variations du vecteur normal. Nous énonçons la définition arithmétique d'un plan discret de façon plus formelle.

Définition 12 *Un ensemble de points $S = \{p^1, \ldots, p^n\}$ de $\mathbb{Z}^3$ correspond à un morceau de* plan discret *de vecteur normal $N = (N_1, N_2, N_3) \in \mathbb{Z}^3$ si pour tout point p^i de S, $1 \leq i \leq n$, nous avons la double inéquation diophantienne :*

$$\mu \leq N \cdot p^i < \mu + \omega \tag{3.1}$$

La valeur ω est appelée épaisseur arithmétique *du plan discret et la valeur μ est appelée la borne inférieure. Le vecteur normal N est irréductible. Dans le cas où ω vaut $||N||_\infty$ (i.e. $\max_{1 \leq i \leq 3} |N_i|$), nous obtenons un* plan discret naïf. *Si ω vaut $\sum_{i=1}^{3} |N_i|$, nous obtenons un* plan discret standard.

Nous pouvons également considérer des plans discrets plus épais en vue de polyédrisations plus grossières. Par la suite, nous nous intéressons plus particulièrement à la reconnaissance de plans discrets naïfs. Nous pouvons considérer sans perte de généralité le

cas où les plans discrets sont une fonction de (x_1, x_2) vers $\mathbb{Z}$. En d'autres termes, nous supposons que la projection des points de S sur le plan est injective. Dans ce cas, le troisième axe est donc appelé *axe majeur*. Les autres cas peuvent être déduits par symétrie. Par conséquent, nous supposons que $||N||_\infty$ est égal à $|N_3|$. De plus, nous supposons que la valeur N_3 est strictement positive. Avec ces hypothèses, le vecteur N est toujours colinéaire au vecteur N' de la forme $(N_1/N_3, N_2/N_3, 1)$ tel que $||N'||_\infty$ vale 1. En utilisant ces notations, nous pouvons reformuler la définition d'un plan discret naïf de la façon suivante :

Définition 13 *L'ensemble de points $S = \{p^1, \ldots, p^n\}$ de $\mathbb{Z}^3$ correspond à un morceau de* plan discret naïf *de vecteur normal $N' \in \mathbb{R}^3$ si pour tout point p^i de S, $1 \leq i \leq n$, nous avons la double inéquation :*

$$\gamma \leq N' \cdot p^i < \gamma + 1 \tag{3.2}$$

où N' est de la forme $(N'_1, N'_2, 1)$ avec N'_1 et N'_2 à valeur dans $[-1, 1]$.

D'après les hypothèses précédentes, nous pouvons donc considérer uniquement les vecteurs normaux dont la norme à l'infini correspond à la valeur absolue de la dernière composante, cette dernière composante valant 1.

Une autre façon de définir les plans discrets consiste à utiliser la définition de la *fonction épaisseur* par rapport à l'ensemble S. Afin de définir cette fonction, nous commençons par aborder la notion de *plan support*. La définition analogue des droites supports en 2D est donnée dans [dBSvKO00].

Définition 14 *Un plan P est dit* support *d'un ensemble de points S de $\mathbb{Z}^3$ si $P \cap S \neq \emptyset$ et S est entièrement contenu dans un des deux hyperplans bornés par P.*

Deux plans supports parallèles *englobent* un ensemble de points si l'ensemble de points est contenu entièrement entre les deux plans et si l'intersection de chaque plan avec l'ensemble de points est non-vide. Notons que deux plans supports englobent un ensemble de points si et seulement si ils englobent l'enveloppe convexe de ces points. L'intersection de chaque plan support avec l'ensemble de points étant non-vide, chaque plan support est confondu avec un sommet, une arête ou une face de l'enveloppe convexe. Soient deux plans supports parallèles P_1 et P_2 englobant un ensemble de points 3D noté S. Si P_1 et P_2 contiennent chacun un sommet de $\text{conv}(S)$, ces sommets forment une paire de *sommets opposés* (ou antipodaux) notée paire S-S, pour Sommet-Sommet. De la même manière, nous définissons les paires : A-A pour Arête-Arête, F-F pour Face-Face, S-F pour Sommet-Face, S-A pour sommet-Arête et A-F pour Arête-Face.

Soit $u = (u_1, u_2)$ un vecteur bi-dimensionnel de $[-1, 1]^2$. Le vecteur tri-dimensionnel N_u correspond au vecteur de la forme $(u_1, u_2, 1)$. Nous définissons à présent la fonction épaisseur.

Définition 15 *La* fonction épaisseur *de $[-1, 1]^2$ vers $\mathbb{R}^+$ par rapport à l'ensemble S est définie de la manière suivante :*

$$ep_S(u) = \max_{1 \leq i \leq n} (N_u \cdot p^i) - \min_{1 \leq i \leq n} (N_u \cdot p^i) \tag{3.3}$$

La fonction épaisseur au point $u \in [-1, 1]^2$ calcule la distance, par rapport au troisième axe, entre les deux plans supports de vecteur normal N_u englobant les points de S (voir [BCK07]).

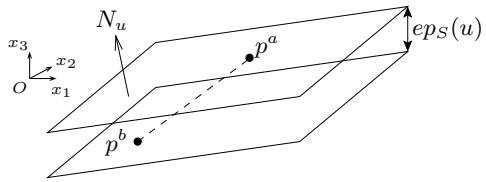


FIG. 3.3 – Définition de la fonction épaisseur

Proposition 4 *L'ensemble S correspond à un morceau de plan discret naïf s'il existe au moins un vecteur u de $[-1,1]^2$ qui vérifie $ep_S(u) < 1$.*

Preuve 5 *D'après (3.3), deux points de S sont associés au produit scalaire maximum et minimum intervenant dans l'expression de la fonction épaisseur à un point donné. Notons p^a et p^b les deux points tels que $ep_S(u) = N_u \cdot (p^a - p^b)$, pour u donné. Comme $N_u \cdot p^a$ et $N_u \cdot p^b$ correspondent respectivement à une valeur maximum et à une valeur minimum, nous en déduisons que la double inégalité $N_u \cdot p^b \leq N_u \cdot p^i \leq N_u \cdot p^a$ est vérifiée pour tout point p^i de S. Si $ep_S(u)$ est strictement inférieur à 1, nous avons $N_u \cdot p^b \leq N_u \cdot p^i < N_u \cdot p^b + 1$. D'après la définition 13, ceci signifie que S correspond à un plan discret naïf.*

Par définition de la fonction épaisseur, le calcul de la valeur $ep_S(u)$ implique le calcul de n produits scalaires. Nous reviendrons plus en détail sur les propriétés de la fonction épaisseur. La figure 3.3 en illustre la définition.

3.3 Quelques Algorithmes de Référence

Depuis le début des années 80, de nombreux algorithmes de reconnaissance de plans discrets ont vu le jour (voir [BCK07]). Ces approches utilisent généralement la programmation linéaire [Buz03, Meg84, ST91, BD07], les enveloppes convexes et la géométrie algorithmique [DR95, DRR94, PS85, VC00, Kim84, KS91, ST91], l'optimisation combinatoire [DR95, DRR94, FST96, GDRZ05, Rev95, VC00] ou encore la propriété de régularité (*evenness property*) [Vee94, Vee93]. Nous décrivons brièvement dans cette section quelques uns des algorithmes de reconnaissance de plans discrets. Nous commençons par introduire les méthodes utilisant la géométrie algorithmique dans la section 3.3.1. Ces méthodes passent généralement par le calcul de l'enveloppe convexe de S. Par la suite, nous abordons dans la section 3.3.2 les méthodes mettant en oeuvre la programmation linéaire. À cette occasion, nous rappelons brièvement les méthodes les plus connues pour la résolution d'un problème de PL et nous décrivons les différentes façons de formuler le problème de reconnaissance de plans discrets en vue d'une résolution par PL. Nous finissons en introduisant des méthodes n'entrant dans aucunes des deux premières catégories comme la méthode utilisant le critère de régularité dans la section 3.3.3. Pour décrire ce panel de méthodes, nous nous sommes largement inspirés des états de l'art établis dans [DR95] et dans [BCK07].

3.3.1 Méthodes Utilisant la Géométrie Algorithmique

Algorithme de Kim

Le premier algorithme utilisant la géométrie algorithmique pour résoudre le problème de reconnaissance de plans discrets est du à Kim [Kim84]. Nous citons ici cette méthode

à titre historique car il s'avère qu'un des arguments sur lesquels se base Kim est faux. Son algorithme n'est donc pas valable mais l'idée de départ mérite d'être présentée. Kim propose et prouve le théorème 4. Ce théorème constitue la base de sa méthode.

Théorème 4 *Un ensemble de points S de $\mathbb{Z}^2$ correspond à un morceau de plan discret si et seulement si il existe une face de l'enveloppe convexe de S telle que la distance euclidienne suivant l'axe majeur entre la face et les points de S est inférieure à* 1.

Ce théorème est équivalent à la proposition 4 qui indique qu'un ensemble de points S correspond à un morceau de plan discret naïf s'il est contenu entre deux plans supports parallèles, séparés d'une distance d'au plus 1 suivant l'axe majeur. Kim part de l'hypothèse qu'un de ces plans supports est forcément porté par une face de l'enveloppe convexe de l'ensemble S. Sa méthode consiste donc à calculer l'enveloppe convexe de l'ensemble de points S, puis pour chaque face, de déterminer la distance entre la face et les points de l'ensemble S. Si pour une face de l'enveloppe convexe la distance maximale par rapport à l'ensemble de points S est strictement inférieure à 1, alors Kim en déduit que l'ensemble S correspond à un morceau de plan discret. L'algorithme 10 résume cette méthode. Notons qu'il renvoie une valeur booléenne afin d'indiquer si l'ensemble de points S correspond à un morceau de plan discret.

Algorithme 10 : Algorithme de Kim

```
RECONNAISSANCE DE PLAN (S : ensemble de n points de Z^3)
Valeur retournée : booléen
1  C ← EnveloppeConvexe(S)
2  EstUnPlan ← faux
3  POUR CHAQUE face f de C
4    SI EstUnPlan
5      Retourner vrai
6    SINON
7      EstUnPlan ← vrai
8    POUR CHAQUE point p de S
9      SI dist(f, p)≥ 1
10       EstUnPlan ← faux
11       Aller ligne 3
12 Retourner EstUnPlan
```

La construction de l'enveloppe convexe 3D se fait en $O(n \log n)$ [PH77] où n correspond au nombre de points de l'ensemble S. De plus, il est nécessaire de tester pour chaque face de l'enveloppe convexe les n points de l'ensemble S. Le nombre de faces étant de l'ordre de $O(n)$, la complexité de la seconde boucle est $O(n^2)$. L'algorithme s'exécute donc en $O(n^2)$. Cette méthode a été améliorée par la suite dans [KS91], ramenant sa complexité à $O(n \log n)$. Malheureusement, l'argument principal de cette méthode est en réalité faux. En effet, Kim suppose qu'un des deux plans supports définissant la distance minimale est forcément porté par une face de l'enveloppe convexe de S. Or, il se peut que les plans supports recherchés soient tous les deux portés par des arêtes de l'enveloppe convexe (voir [DR95, BCK07]). Sa méthode n'est donc pas viable car elle ne considère pas tous les plans supports nécessaires. Nous avons tout de même choisi de présenter cet algorithme à titre historique.

Dans [PBDR06], Provot et al. s'intéressent à des plans discrets plus généraux. En effet, au lieu de considérer uniquement des morceaux de plans discrets naïfs ou standards au sens de la définition 12, ils s'intéressent à la reconnaissance de morceaux de plans discrets *épais*. Soit S un morceau de plan discret caractérisé par la double inéquation $\mu \leq ax + by + cz < \mu + \omega$, S est appelé *morceau de plan discret flou d'épaisseur* v si $(\omega - 1)/N(a, b, c) \leq v$ où N représente une norme. Si la norme choisie correspond à la norme euclidienne, alors $(\omega - 1)/N(a, b, c)$ correspond à la distance euclidienne entre les deux plans supports de S de vecteur normal (a, b, c). Si la norme choisie correspond à la norme à l'infini, alors $(\omega - 1)/N(a, b, c)$ correspond à la distance entre les deux plans supports de S de vecteur normal (a, b, c) suivant l'axe majeur. Pour déterminer si un ensemble de points S correspond à un morceau de plan discret flou d'épaisseur v, il s'agit donc de déterminer le quintuplet (a, b, c, μ, ω) tel que tout point (x, y, z) de S vérifie $\mu \leq ax+by+cz < \mu+\omega$ et tel que $(\omega-1)/N(a, b, c)$ est minimal. L'ensemble S correspond à un morceau de plan discret flou d'épaisseur v si et seulement si $(\omega - 1)/N(a, b, c)$ est inférieur ou égal à v. Dans [HT88], Houle et Toussaint montrent que, lorsque la norme choisie est la norme euclidienne, cette épaisseur minimale est atteinte pour deux plans supports de S vérifiant :
(a) Chacun des plans supports passe par une arête de l'enveloppe convexe de S définissant ainsi une paire A-A

ou

(b) Un des plans supports passe par un sommet de conv(S) et l'autre passe par une face de conv(S) définissant ainsi une paire S-F

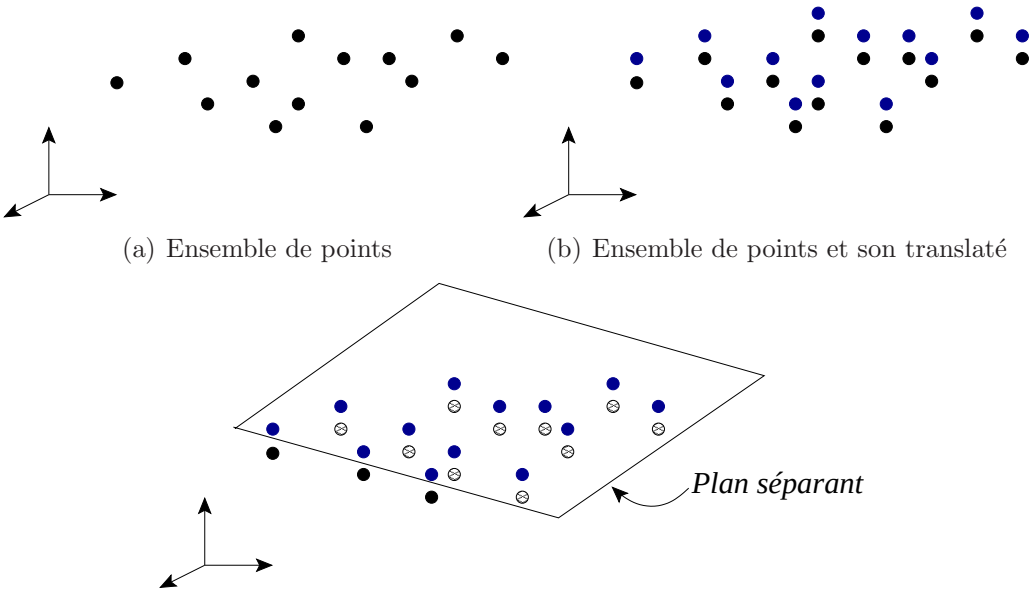
En 2001, Gärtner et Herrmann proposèrent dans [GH01] une méthode inspirée de l'algorithme du *rotating calipers* de Toussaint [Tou83] pour déterminer l'ensemble des plans supports de S vérifiant une des deux conditions (a) ou (b). Cette méthode revient donc à déterminer l'ensemble des paires A-A et S-F de l'enveloppe convexe de l'ensemble S. Leur méthode consiste à calculer tout d'abord l'enveloppe convexe de l'ensemble S. Nous choisissons alors une face f de conv(S) et nous déterminons l'ensemble de ses sommets opposés. Nous trouvons ainsi une première paire S-F et les plans supports correspondants P_1 et P_2. Il s'agit ensuite de faire tourner le plan P_2 contenant f autour d'une arête e bordant la face f, tout en gardant P_1 et P_2 parallèles, jusqu'à ce que P_2 rencontre une autre face f' de conv(S). Nous déterminons ainsi toutes les arêtes opposées à e et les sommets opposés à f'. Cette méthode est réitérée pour chaque arête de conv(S) afin de déterminer toutes les paires A-A et V-F. Cette méthode s'exécute en $O(n^2)$ où n correspond au nombre de points de l'ensemble S. Provot et al. propose une version incrémentale de cette méthode dans [PBDR06] en $O(n^3)$.

Algorithme Basé sur la Séparabilité

Une autre méthode consiste à utiliser un test de séparabilité. Elle se base sur le théorème suivant [ST91], illustré en figure 3.4.

Théorème 5 *Soit S' l'ensemble de points S translaté de 1 suivant l'axe majeur. L'ensemble S correspond à un morceau de plan discret si S et S' peuvent être séparés par un plan euclidien.*

Une première option consiste à calculer l'enveloppe convexe des ensembles S et S' et de vérifier si les deux enveloppes convexes peuvent être séparées par un plan euclidien [ST91]. Notons que ceci est possible si et seulement si l'intersection des deux enveloppes



(a) Ensemble de points (b) Ensemble de points et son translaté

(c) Plan séparant

FIG. 3.4 – Séparabilité des ensembles

est vide. Dans ce cas, S correspond à un morceau de plan discret. L'algorithme 11 résume cette méthode.

Algorithme 11 : Algorithme de reconnaissance par séparabilité des enveloppes

RECONNAISSANCE DE PLAN (S : ensemble de n points de $\mathbb{Z}^3$)
Valeur retournée : booléen
1 Déterminer l'axe majeur
2 $S' \leftarrow$Translaté de S suivant l'axe majeur d'un pas de 1
3 $C \leftarrow$ EnveloppeConvexe(S)
4 $C' \leftarrow$ EnveloppeConvexe(S')
5 SI C et C' s'intersectent
6 Retourner faux
7 SINON
8 Retourner vrai

Le calcul d'enveloppes convexes en 3D se fait en $O(n \log n)$ [PH77] de même que la détermination de l'intersection des polyèdres [DM80, Dye80] où n correspond au nombre de points de l'ensemble S. L'algorithme de reconnaissance de plans discrets mettant en oeuvre la séparabilité des enveloppes convexes s'exécute donc en $O(n \log n)$.

La seconde option utilise des techniques de programmation linéaire et consiste à associer à chaque point de S et de S' une inégalité. En effet, soit P un plan d'équation $ax + by + cz + d = 0$, le plan P est séparant si tout point (p_1, p_2, p_3) de S vérifie $ap_1+bp_2+cp_3+d \leq 0$ et si tout point (p_1, p_2, p_3+1) de S' vérifie $ap_1+bp_2+c(p_3+1)+d > 0$. Le problème revient alors à résoudre un système de $2n$ inéquations à trois variables. Cependant, les méthodes de programmation linéaire sont exposées dans la section suivante.

Algorithme des Cordes

Enfin, le dernier algorithme présenté dans cette section est l'algorithme des cordes proposé par Gérard dans [GÓ3] et repris ensuite dans [GDRZ05]. Cette méthode utilise l'*ensemble des cordes* associé à S ansi que l'*épaisseur géométrique* de l'ensemble S. Ces deux notions sont définies ci-dessous. L'ensemble des cordes est illustré par la figure 3.5.

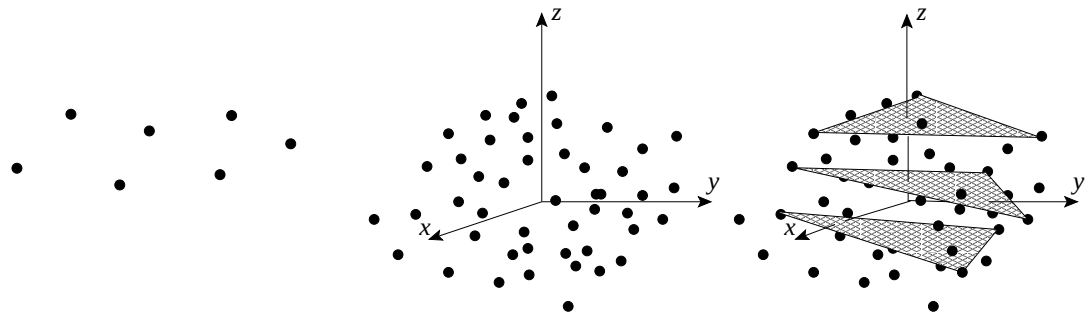
Définition 16 *L'*ensemble des cordes *de S, noté cordes(S), est l'ensemble des vecteurs définis par deux points de l'ensemble S : cordes$(S) = S + (-S) = \{p - p' | p, p' \in S\}$.*

Définition 17 *Soit x_3 l'axe majeur associé à l'ensemble S, l'*épaisseur géométrique *de S correspond à la troisième coordonnée du point d'intersection de l'enveloppe convexe des cordes de S (noté conv(cordes(S))) avec l'axe x_3.*

L'algorithme des cordes utilise alors le théorème suivant [GDRZ05] :

Théorème 6 *L'ensemble S correspond à un morceau de plan discret si et seulement si son épaisseur géométrique est strictement plus petite que* 1.

Soit (a, b, c) le vecteur normal de la face de conv(*cordes*(S)) qui détermine l'épaisseur géométrique. Si l'épaisseur géométrique de S est strictement inférieure à 1, alors les coordonnées du vecteur (a, b, c) correspondent aux coefficients du plan définissant le morceau de plan discret (i.e. $\forall (x, y, z) \in S, h \leq ax+by+cz < h+c$). Le but de l'algorithme est donc de déterminer cette face. Néanmoins, procéder à un calcul d'enveloppe convexe en 3D est très coûteux d'autant plus que l'ensemble des cordes comporte un nombre quadratique de points par rapport à l'ensemble de points initial. C'est pourquoi cette méthode recherche plutôt un triangle à sommets portés par les points de $conv(S)$ confondu avec la face de l'enveloppe convexe recherchée. Nous qualifions ce triangle de *plus haut triangle* de l'ensemble des cordes. Pour le déterminer, l'algorithme initialise un triangle à sommets portés par trois points non-colinéaires de *cordes*(S) traversé par l'axe vertical. Soit N le vecteur normal de ce triangle, soient p_{min} et p_{max} les points tels que $N \cdot p_{min} = \min_{p \in cordes(S)} N \cdot p$ et $N \cdot p_{max} = \max_{p \in cordes(S)} N \cdot p$. Le point $M = p_{max} - p_{min}$ est calculé. Si le point M appartient au triangle, alors le triangle est le plus haut triangle recherché. Sinon, ce point et le triangle forment un tétraèdre dont une des faces est plus haute que le triangle de référence et l'algorithme réitère (voir la figure 3.5). L'algorithme 12 présente la partie du programme qui recherche le plus haut triangle dans l'espace des cordes.



(a) Ensemble de points S (b) Ensemble *cordes*(S) (c) Triangles de *cordes*(S)

FIG. 3.5 – Ensemble des cordes

Algorithme 12 : Algorithme des cordes : trouver le plus haut triangle

```
PLUS HAUT TRIANGLE(S : ensemble de n points de Z^3)
Valeur retournée : triangle
1  T ←TriangleInitial(cordes(S))
2  continue ← vrai
3  TANT QUE continue
4    N ← VecteurNormal(T)
5    M ← DifférenceMinMax(S, N)
6    SI M est strictement au-dessus de T
7      T ← PlusHautTriangleTétraèdre(T, M)
8    SINON
9      continue ← faux
10 Retourner T
```

La complexité de cet algorithme dans le pire cas est $O(n^7)$ où n correspond au nombres de points de S. En effet, le nombre total de triangles dans l'espace des cordes est de l'ordre de $O(n^6)$ ce qui correspond au nombre maximum de passages dans la boucle principale. De plus, chaque boucle s'exécute en $O(n)$. Cependant, cette borne de complexité est rarement atteinte et le comportement de l'algorithme semble linéaire.

3.3.2 Méthodes Basées sur la Programmation Linéaire

Formulation d'un Problème de Programmation Linéaire

En mathématiques, les problèmes de *programmation linéaire*, couramment notés PL, consistent à optimiser une fonction objectif linéaire sous un système de contraintes elles-mêmes linéaires. Le système de contraintes définit un *domaine de réalisabilité*. Le domaine de réalisabilité correspond à l'ensemble des points qui satisfont le système de contraintes et qui sont donc des candidats potentiels pour le problème d'optimisation. Si le domaine de réalisabilité est vide, alors le problème d'optimisation n'admet pas de solution et nous disons alors qu'il n'est pas réalisable. Notons qu'on distingue les problèmes d'optimisation et les problèmes de réalisabilité. Un problème de réalisabilité consiste à déterminer si le domaine de réalisabilité défini par un système de contraintes est non-vide alors que le problème d'optimisation consiste à trouver l'optimum d'une fonction objectif sous un système de contraintes.

Considérons un problème de programmation linéaire visant à minimiser une fonction objectif. Soient A une matrice à n lignes et d colonnes, c et x des vecteurs colonnes de taille d et b un vecteur colonne de taille n. Ce problème peut également être écrit sous sa *forme standard* avec A' la matrice $[A|\text{Identité}(n \times n)]$ et c' la matrice $[c|\text{Zéro}(n)]$. Le problème de programmation linéaire peut être formulé de manière classique (à gauche) ou sous sa forme standard (à droite) :

$$\begin{array}{lll} \text{Minimiser} & c^T \cdot x & \\ \text{avec} & A \cdot x \geq b & \text{ou} \\ & x \geq 0 & \end{array} \qquad \begin{array}{lll} \text{Minimiser} & {c'}^T \cdot x' & \\ \text{avec} & A' \cdot x' = b & \\ & x' \geq 0 & \end{array} \tag{3.4}$$

De plus, il est prouvé qu'à tout problème de PL est associé un problème dual et que si un des deux problèmes est réalisable et admet un optimum alors il en est de même pour l'autre. En outre, les valeurs atteintes pour l'optimum du problème primal et du

problème dual sont identiques. Si le problème primal est un problème de minimisation sous contraintes de la forme "supérieur ou égal" alors le problème dual associé est un problème de maximisation sous contraintes de la forme "inférieur ou égal" et vice versa. De plus, à toute contrainte du primal nous associons une variable dans le dual et à toute variable du primal nous associons une contrainte dans le dual. Le problème primal et le problème dual peuvent être formulés de la manière suivante :

$$\text{Primal} \begin{cases} \text{Minimiser} & c^T \cdot x \\ \text{avec} & A \cdot x \geq b \\ & x \geq 0 \end{cases} \qquad \text{Dual} \begin{cases} \text{Maximiser} & b^T \cdot \lambda \\ \text{avec} & A^T \cdot \lambda \leq c \\ & \lambda \geq 0 \end{cases} \tag{3.5}$$

Résolution d'un Problème de Programmation Linéaire

Pour résoudre les problèmes de programmation linéaire, différentes techniques peuvent être envisagées. Les résultats optimaux de Megiddo [Meg83, Meg84], très connus en PL, énoncent qu'un système de m contraintes linéaires dans $\mathbb{R}^d$ peut être résolu en $O(m)$. La méthode de Megiddo pour déterminer l'optimum d'un système de contraintes linéaires consiste à coupler les contraintes deux à deux et à éliminer une partie des contraintes du système grâce à un critère de suppression. Le critère de suppression utilisé est le suivant. Les contraintes sont séparées en deux familles : les contraintes de la forme $x_d \geq ax_1 + bx_2 + \cdots$ et les contraintes de la forme $x_d \leq ax_1 + bx_2 + \cdots$. Pour deux contraintes de la même famille dont les hyperplans correspondants ne sont pas parallèles, il existe un hyperplan vertical séparant les deux contraintes en leur intersection. Si les solutions optimales sont situées d'un côté du plan séparant alors une des deux contraintes est éliminée. À chaque itération, un pourcentage minimum de contraintes est éliminé et l'algorithme réitère jusqu'à ce que le nombre de contraintes devienne constant. Chaque itération s'exécutant en $O(m)$, l'algorithme est linéaire suivant le nombre de contraintes du système (voir [Buz02] pour une étude plus détaillée des résultats de Megiddo). Notons qu'il appartient à la classe des méthodes fortement polynomiales, c'est-à-dire dont la complexité ne dépend que de n et de d à l'inverse de la méthode de Karmakar [Kar84], par exemple, dont la complexité dépend du nombre de bits nécessaires pour représenter les valeurs engagées dans les calculs.

Cependant, l'algorithme de Megiddo est trop complexe pour être utilisé en pratique car il réserve une multitude de cas particuliers. De plus, il est doublement exponentiel en la dimension puisqu'en réalité sa complexité est en $O(2^{2^d} n)$ et s'avère assez lent. Il n'est donc pas utilisé en PL. D'autres méthodes efficaces de programmation linéaire peuvent être envisagées comme l'algorithme du simplexe proposée par Dantzig en 1947 dans [Dan98] (voir aussi [Chv83]). La méthode du simplexe est fondamentale en programmation linéaire. Elle consiste à optimiser une fonction objectif sous un système de contraintes en avançant sur les arêtes du polytope défini par le système de contraintes. Chaque déplacement le long des arêtes du polytope mène à un nouveau sommet pour lequel la fonction objectif est améliorée ou inchangée. Cependant, malgré l'efficacité en pratique de cette méthode, ce qui lui vaut sa popularité, sa complexité dans le pire cas n'est pas polynomiale. En effet, il est possible d'exhiber des cas pathologique à n variables et $2n$ contraintes qui nécessitent un nombre exponentiel de $2^n - 1$ itérations.

Pour finir avec la description des méthodes de programmation linéaire, nous présentons

en quelques lignes la méthode de Fourier-Motzkin [Keß96, Sch98], dûe à Joseph Fourier et Theodore Motzkin. Cette méthode est souvent utilisée pour montrer la réalisabilité d'un système de contraintes linéaires comportant un faible nombre de contraintes. L'algorithme élimine au fur et à mesure des variables du système. À chaque variable éliminée, l'algorithme crée un nouveau système de contraintes équivalent au premier sur les variables restantes. Lorsque toutes les variables sont éliminées, nous obtenons un système d'inégalités constantes qu'il suffit de vérifier. Il est prouvé que si toutes les inégalités du système final sont vraies, alors le système initial est réalisable. Soit x_d la variable à éliminer, nous ramenons les contraintes du système à des contraintes de la forme $\sum_{i=1}^{d-1} a_i x_i \leq x_d$ ou $\sum_{i=1}^{d-1} a'_i x_i \geq x_d$ selon le signe associé à la variable x_d dans chaque contrainte. Les contraintes de la première forme sont alors couplées avec les contraintes de la deuxième forme. Nous obtenons alors des doubles inégalités du type $\sum_{i=1}^{d-1} a_i x_i \leq x_d \leq \sum_{i=1}^{d-1} a'_i x_i$ qu'on ramène à $\sum_{i=1}^{d-1} a_i x_i \leq \sum_{i=1}^{d-1} a'_i x_i$, ce qui élimine la variable x_d. Le nombre de contraintes du nouveau système est quadratique dans le pire cas suivant le nombre de contraintes du système précédent. La complexité de l'algorithme est donc exponentielle. Son utilisation est réservée aux systèmes à faible nombre de contraintes.

Formulation du Problème de Reconnaissance de Plans Discrets

En 1991, Stojmenovic met en évidence que le problème de reconnaissance de plans discrets correspond à un problème de PL et il montre pour la première fois dans [ST91] que ce même problème peut être résolu en temps linéaire grâce aux résultats de Megiddo [Meg83, Meg84]. En fait, le problème de reconnaissance de plans discrets peut être formulé comme un problème de PL de plusieurs façons différentes. En effet, il peut être perçu comme un problème de réalisabilité sous un système de contraintes ou comme un problème d'optimisation. Nous décrivons dans cette section les deux formulations du problème de reconnaissance de plans discrets les plus intuitives en PL.

Première Formulation (problème de réalisabilité) **:** D'après la définition 13 des plans discrets, lorsque l'ensemble de n points S correspond à un morceau de plan discret alors chaque point p^i, $1 \leq i \leq n$, de S doit satisfaire deux inégalités. Ces inégalités sont de la forme $N' \cdot p^i - \Gamma \geq 0$ et $N' \cdot P^i - \Gamma - 1 < 0$, où $N' = (N'_1, N'_2, 1)$ correspond au vecteur normal du plan discret et vérifie $||N'||_\infty = 1$. Le problème de reconnaissance de plans discrets peut donc directement être vu comme un problème de réalisabilité sur un système de $2n$ contraintes linéaires à deux inconnues (la dernière coordonnée de N' étant fixée à 1). Pour cela, nous pouvons utiliser une des techniques de programmation linéaire décrites dans la section 3.3.2 (voir [Sch98]).

Deuxième Formulation (problème d'optimisation) **:** Dans [PBDR06], Provot et al. introduisent la définition des morceaux de plans discrets flous d'épaisseur v que nous avons rappelée dans la section 3.3.1. Pour déterminer si un ensemble de n points de $\mathbb{Z}^3$ noté S correspond à un morceau de plan discret flou d'épaisseur v, nous cherchons à déterminer deux plans P et P' supports de S de la forme $ax + by + cz + \mu = 0$ et $ax + by + cz + \mu + \omega - 1 = 0$ tels que $(\mu - 1)/N(a, b, c)$ soit minimal. Si $(\mu - 1)/N(a, b, c)$ est inférieur à v, alors S correspond à un morceau de plan discret flou d'épaisseur v.

Supposons que l'axe majeur de l'ensemble de points S correspond au troisième axe. Si la norme choisie pour la définition des plans discrets flous est la norme à l'infini, alors l'expression $(\mu - 1)/N(a, b, c)$ est égale à $(\mu - 1)/|c|$. Posons $\alpha = a/c$, $\beta = b/c$,

$h = \mu/c$ et $e = (\omega - 1)/c$, les deux plans supports recherchés sont donc de la forme $P : z(x, y) = \alpha x + \beta y + h$ et $P' : z(x, y) = \alpha x + \beta y + h + e$ et nous cherchons à minimiser la valeur e. Notons que comme $||(a, b, c)||_\infty$ est égale à c, alors α et β sont compris entre -1 et 1. À chaque point (x_i, y_i, z_i) de S, $1 \leq i \leq n$, nous associons donc deux inégalités : $\alpha x_i + \beta x_i + h \leq z_i$ et $\alpha x_i + \beta x_i + h + e \geq z_i$. Le problème d'optimisation est donc de la forme :

$$\begin{array}{l} \text{Minimiser } e \\ \\ \left\{ \begin{array}{l} -\alpha x_i - \beta x_i - h \geq -z_i \\ \alpha x_i + \beta x_i + h + e \geq z_i \\ i = 1 \cdots n \end{array} \right. \\ \\ |\alpha| \leq 1, |\beta| \leq 1 \\ \alpha, \beta, h \in \mathbb{R}, e \geq 0 \end{array} \tag{3.6}$$

Pour résoudre ce problème d'optimisation, les auteurs de [PBDR06] proposent d'utiliser l'algorithme du simplexe sur le problème dual afin de manipuler des matrices plus petites. Dans ce cas, le problème dual de (3.6) en forme standard s'écrit de la manière suivante :

$$\begin{array}{l} \text{Maximiser } [-z_1 \cdots - z_n | z_1 \cdots z_n | -1 -1 -1 -10] \cdot \lambda \\ \\ \begin{bmatrix} -x_1 & \cdots & -x_n & x_1 & \cdots & x_n & -1 & 1 & 0 & 0 & 0 \\ -y_1 & \cdots & -y_n & y_1 & \cdots & y_n & 0 & 0 & -1 & 1 & 0 \\ -1 & \cdots & -1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \cdots \\ \lambda_{2n+5} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ \\ \lambda \geq 0 \end{array} \tag{3.7}$$

L'algorithme du simplexe est donc utilisé pour résoudre le problème dual. Les auteurs de [PBDR06] montrent qu'à chaque itération la valeur de e décroît strictement. L'algorithme du simplexe ne cycle donc pas. Notons que dans le cas général, la méthode du simplexe peut considérer plusieurs fois les mêmes points de l'espace de recherche, ce qui entraine sa complexité exponentielle. Dans notre cas, le nombre d'itérations de l'algorithme est donc borné par le nombre de plans supports considérés, soit au plus C^4_{2n+5}. La complexité de la méthode est donc $O(k^4)$ avec k le nombre de sommets de l'enveloppe convexe de S.

Algorithme de la Pré-Image

Le problème de reconnaissance de plans discrets peut également être résolu grâce à des algorithmes calculant la pré-image [VC00, CSD+03, Coe02, DA09]. Ces algorithmes mettent également en oeuvre la programmation linéaire bien que la formulation du problème s'avère différente de celles énoncées précédemment dans cette section. Soit $f : X \rightarrow Y$ une fonction, la pré-image de $y \in Y$ correspond à l'ensemble des points dont l'image est y, c'est-à-dire $\{x \in X | f(x) = y\}$. Pour le problème de reconnaissance de plans discrets, la pré-image est définie comme suit.

Définition 18 *Soit S un ensemble de points correspondant à un morceau de plan discret, la* pré-image *de l'ensemble S est l'ensemble des plans euclidiens dont la discrétisation*

contient S.

Cette pré-image correspond à l'ensemble des solutions réalisables du systèmes d'inégalités induit par la définition des plans discrets. Elle correspond à un polyèdre à sommets rationnels dans l'espace des paramètres et peut être écrite comme $\{(\alpha, \beta, \gamma) \in [0,1]^2 \times [0,1[\,|\,\forall (x,y,z) \in S, z \leq \alpha x + \beta y + \gamma < z + 1\}$. Le but de ce type d'algorithme est de construire la pré-image associée à l'ensemble de points S. Si la pré-image correspond à un polyèdre vide, alors S ne correspond pas à un morceau de plan discret. Si la pré-image contient au moins un point, alors S correspond à un morceau de plan discret. Pour cet algorithme incrémental, la pré-image est initialisée à $\mathbb{R}^3$. À chaque point de S sont associées deux contraintes linéaires, et chacune de ces contraintes fournit un moyen de réduire la pré-image courante par une coupe. L'algorithme se termine lorsque la pré-image est vide ou lorsque toutes les contraintes ont été utilisées. L'algorithme 13 résume cette méthode.

Algorithme 13 : Algorithme de la pré-image

```
RECONNAISSANCE-PAR-PRÉ-IMAGE(S : ensemble de n points de Z^3)
Valeur retournée : booléen
1  PreImage ← R^3
2  POUR CHAQUE point p de S FAIRE
3    Soit C_1 et C_2 les contraintes associées à p
4    POUR CHAQUE contrainte C de {C_1, C_2} FAIRE
5      Coupe(PreImage, C)
6      SI PreImage = ∅
7        Retourner faux
8  Retourner vrai
```

L'intersection d'un plan euclidien et d'un polyèdre peut être calculée en temps linéaire suivant le nombre de sommets du polyèdre. Dans [CSD+03], les auteurs indiquent que le nombre de sommets de la pré-image est bornée à chaque appel par le nombre de sommets de l'enveloppe convexe de l'ensemble de points courant. L'algorithme de reconnaissance s'exécute donc en $O(nE)$ avec E le nombre de sommets maximum de la pré-image au fil des appels récursifs et n le nombre de points de l'ensemble S. Ils ajoutent que la méthode s'avère efficace en pratique. De plus, l'algorithme admet une programmation en ligne s'exécutant en $O(E)$.

3.3.3 Autres Méthodes

La première méthode abordée dans cette section utilise le *critère de régularité*. Le critère de régularité ou *evenness property* a été proposé en dimension quelconque par Veelaert [Vee94, Vee94] d'après la définition 2D donnée par Hung [Hun85].

Définition 19 *Soit S un ensemble de $\mathbb{Z}^3$, S est dit* régulier *si et seulement si sa projection suivant la direction x_3 est bijective et si pour tout quadruplet (A, B, C, D) de points de S tel que $(A_1 - B_1, A_2 - B_2) = (C_1 - D_1, C_2 - D_2)$ nous avons bien $|(A_3 - B_3) - (C_3 - D_3)| \leq 1$.*

Veelaert montre qu'un rectangle de voxels correspond à un morceau de plan discret si et seulement si il est régulier. Il conçoit alors un algorithme de reconnaissance de morceaux de plans discrets rectangulaires qui consiste à vérifier ce critère. L'algorithme 14 résume

cette méthode. Dans cet algorithme, T_{min} et T_{max} représentent des sortes de tableaux bi-dimensionnel dont nous initialisons les cases $[i][j]$ uniquement s'il existe deux points a et b de S tels que $i = a_x - b_x$ et $j = a_y - b_y$. Ces cases de T_{min} et T_{max} sont respectivement initialisées à ∞ et à $-\infty$. Nous stockons ensuite dans la case $[i][j]$ de T_{min} la plus petite coordonnée z des vecteurs ba, avec $a, b \in S$, tels que $a_x - b_x = i$ et $a_y - b_y = j$. De la même manière, nous stockons dans la case $[i][j]$ de T_{max} la plus grande coordonnée z des vecteurs ba, avec $a, b \in S$, tels que $a_x - b_x = i$ et $a_y - b_y = j$. Il suffira alors de s'assurer que $T_{max}[i][j] - T_{min}[i][j]$ est bien strictement inférieur à 1 pour toutes les cases initialisées.

Algorithme 14 : Algorithme avec le critère de régularité

RECONNAISSANCE-PAR-RÉGULARITÉ(S : ensemble de n points de $\mathbb{Z}^3$)
Valeur retournée : booléen
1 POUR TOUT point a de S
2 POUR TOUT point b de S
3 $Tab_{min}[a_x - b_x][a_y - b_y] \leftarrow \infty$
4 $Tab_{max}[a_x - b_x][a_y - b_y] \leftarrow -\infty$
5 POUR TOUT point a de S
6 POUR TOUT point b de S
7 SI $Tab_{min}[a_x - b_x][a_y - b_y] > a_z - b_z$
8 $Tab_{min}[a_x - b_x][a_y - b_y] \leftarrow a_z - b_z$
9 SI $Tab_{max}[a_x - b_x][a_y - b_y] < a_z - b_z$
10 $Tab_{max}[a_x - b_x][a_y - b_y] \leftarrow a_z - b_z$
11 POUR TOUT point a de S
12 POUR TOUT point b de S
13 SI $Tab_{max}[a_x - b_x][a_y - b_y] - Tab_{min}[a_x - b_x][a_y - b_y] > 1$
14 Retourner faux
15 Retourner vrai

Nous observons que le critère de régularité de l'ensemble S peut être vérifié en $O(n^2)$ où n correspond au nombre de points de l'ensemble S. La complexité de cet algorithme est donc quadratique en le nombre de points de l'ensemble.

D'autres méthodes de reconnaissance de plans discrets peuvent être trouvées dans la littérature mais celles-ci ne s'appliquent pas dans le cas général. Nous nous contentons donc de les citer en cette fin de section. Un de ces algorithme met en oeuvre l'ajustement par la méthode des moindres carrés [KSv96]. Cette méthode s'exécute en temps linéaire. Cependant, comme l'algorithme précédent, cette méthode ne s'applique qu'aux ensembles de points rectangulaires. Dans [Fer09], Fernique propose une méthode de génération et de reconnaissance de plans discrets particuliers dits "en escalier", d'après leur définition donnée dans [Vui98]. Sa méthode utilise la connection entre la théorie des mots et les fractions continues. Il propose un algorithme hybride de reconnaissance de plans discrets en escalier qui combine un algorithme classique de reconnaissance de plans discrets comme l'algorithme de la pré-image, par exemple, avec l'algorithme de Brun. L'algorithme de Brun [Bre81] est une méthode qui exploite les fractions continues multi-dimensionnelles. Malheureusement, la complexité de l'algorithme de Fernique n'a pas été prouvée à ce jour.

3.4 Conclusion

Nous avons présenté dans ce chapitre différentes façons de définir un morceau de plan discret, et en particulier un morceau de plan discret naïf. Nous avons ensuite décrit quelques uns des principaux algorithmes de reconnaissance de plans discrets. Ces méthodes utilisent la programmation linéaire, la géométrie algorithmique ou encore le critère de régularité. Nous remarquons que les seules méthodes qui ont une complexité linéaire optimale dans le cas général utilisent les résultats de Megiddo en programmation linéaire. Malheureusement, ces résultats sont connus pour être difficilement implémentables et d'exécution assez lente en pratique. Quant aux autres algorithmes présentés, il n'atteignent pas la complexité optimale. Notons cependant que l'algorithme des cordes, malgré sa complexité dans le pire cas en $O(n^7)$, est efficace en pratique et semble avoir un comportement linéaire. Le tableau 3.1 reprend les différents algorithmes. Nous rappelons que n désigne le nombre de points de l'ensemble S, k désigne le nombre de sommets de l'enveloppe convexe calculée et E désigne le nombre maximum de sommets de la pré-image.

Algorithme	Complexité	Réf.	Remarques
Algorithme de Kim	$O(n^2)$	[Kim84]	non valide
Rotating calipers 3D	$O(n^3)$	[PBDR06]	
Séparabilité des enveloppes convexes	$O(n \log n)$	[ST91]	
PL et algorithme de Megiddo	$O(n)$	[Meg84]	difficile à implémenter
PL et simplexe	exponentiel	[Chv83]	efficace
PL et Fourier-Motzkin	exponentiel	[Keß96]	
Optimisation de l'épaisseur et simplexe	$O(k^4)$	[PBDR06]	efficace
Algorithme des cordes	$O(n^7)$	[GDRZ05]	efficace
Algorithme de la pré-image	$O(nE)$	[VC00]	efficace
Critère de régularité	$O(n^2)$	[Vee94]	
Moindres carrés	$O(n)$	[KSv96]	ensembles rectangulaires

Tab. 3.1 – Comparatif des algorithmes de reconnaissance de plans discrets

Chapitre 4

Nouvelle Approche Efficace pour la Reconnaissance de Plans Discrets

4.1 Introduction

Comme nous l'avons vu dans le chapitre 3, la reconnaissance de plans discrets est un problème largement étudié en géométrie discrète. Nous décrivons dans ce chapitre une nouvelle méthode pour décider si un ensemble de n points de $\mathbb{Z}^3$ correspond à la discrétisation d'un morceau de plan euclidien. La méthode que nous décrivons ici est celle proposée à la conférence internationale DGCI (Discrete Geometry for Computer Imagery) de 2008 (voir [CB08a]). Une version préliminaire de cet algorithme avait déjà été présentée en 2006 par Lilian Buzer dans [Buz06].

La plupart des méthodes existantes ne combinent pas à la fois une complexité intéressante dans le pire cas et une efficacité en pratique. Par exemple, les méthodes utilisant les résultats optimaux de Megiddo (voir [Meg84]) ont beau être linéaires, elles peuvent difficilement être utilisées en pratique. Quant à la méthode des cordes (voir [GDRZ05]), elle parvient à traiter un ensemble de 10^6 points en parcourant environ 10 fois l'ensemble des cordes mais sa complexité dans le pire cas est de l'ordre de $O(n^7)$. La méthode de 2006 de Lilian Buzer [Buz06] atteint une complexité dans le pire cas de $O(n \log^2 D)$ où $D-1$ représente la plus grande longueur d'une boite englobant l'ensemble de points et elle reconnaît un ensemble dense de 10^6 points en environ 360 itérations linéaires. Notre méthode actuelle est plus efficace en pratique puisqu'elle reconnaît un ensemble dense de 10^6 points en environ 10 itérations linéaires et elle atteint une complexité dans le pire cas de $O(n \log D)$.

La méthode de 2006 et la méthode présentée ici transforment le problème de reconnaissance de plans discrets naïfs en un problème de réalisabilité sur une fonction convexe bi-dimensionelle dite *fonction épaisseur*. Par conséquent, ces méthodes ne prennent en compte que deux paramètres et nous pouvons utiliser des techniques de géométrie discrète planaire afin de déterminer si l'espace des solutions 2D est vide ou pas. Pour résoudre le problème de réalisabilité, notre approche actuelle utilise une propriété du centre de gravité tandis que la méthode de 2006 combine l'oracle de Megiddo et une recherche dichotomique mono-dimensionnelle. Ce principal changement améliore la complexité en temps de la méthode et diminue considérablement son nombre d'itérations. Notre algorithme peut également être conçu de façon incrémentale.

Dans la section 4.2, nous montrons comment le problème de reconnaissance de plans discrets peut être transformé en un problème de réalisabilité sur une fonction convexe bi-

dimensionnelle, la fonction épaisseur. Nous introduisons dans la section 4.3 une méthode pour résoudre ce problème de réalisabilité en utilisant le calcul d'un sous-gradient de la fonction épaisseur. Pour résoudre le problème de réalisabilité, nous réduisons l'espace de recherche continu en appliquant itérativement des coupes jusqu'à trouver un point solution ou jusqu'à ce que l'espace de recherche devienne vide. Dans la section 4.4, nous décrivons cette étape de réduction et nous commentons sont efficacité. Pour améliorer notre algorithme, nous discrétisons le domaine de recherche du problème de réalisabilité. Nous justifions dans la section 4.5 le choix du pas de discrétisation et nous expliquons comment adapter notre méthode à un domaine de recherche discrétisé. Nous analysons la complexité en temps de notre méthode dans la section 4.6. Dans la section 4.7, nous décrivons la version incrémentale de notre algorithme et nous terminons ce chapitre avec quelques résultats expérimentaux et la description d'améliorations pratiques.

4.2 La Reconnaissance de Plans Comme un Problème de Réalisabilité

Dans la section 3.2, nous avons défini la fonction épaisseur par rapport à un ensemble de points de $\mathbb{Z}^3$ noté S comme une fonction de $[-1,1]^2$ vers $\mathbb{R}^+$ définie par :

$$ep_S(u) = \max_{1\leq i\leq n}(N_u \cdot p^i) - \min_{1\leq i\leq n}(N_u \cdot p^i) \tag{4.1}$$

où $u = (u_1, u_2)$ appartient à $[-1,1]^2$ et N_u correspond au vecteur $(u_1, u_2, 1)$.

D'après la proposition 4, l'ensemble de points S correspond à un morceau de plan discret s'il existe un vecteur u de $[-1,1]^2$ qui vérifie $ep_S(u) < 1$. Notre problème de reconnaissance de plans discrets revient donc à déterminer un tel vecteur u ou à montrer qu'aucun vecteur de $[-1,1]^2$ ne satisfait $ep_S(u) < 1$. Par conséquent, nous nous ramènons à un problème de réalisabilité sur la fonction bi-dimensionnelle ep_S.

Proposition 6 *La fonction ep_S est une fonction convexe.*

Preuve 7 *Soit N_u le vecteur associé au vecteur $u \in [-1,1]^2$. Considérons la fonction $g^i(u)$ de $[-1,1]^2$ vers $\mathbb{R}$ qui associe au vecteur u la valeur $N_u \cdot p^i$. Cette fonction est une fonction affine, elle est donc convexe. Remarquons à présent que la fonction $ep_S(u)$ peut être ré-écrite comme $\max_{1\leq i\leq n}(g^i(u)) + \max_{1\leq i\leq n}(-g^i(u))$. Comme le maximum des fonctions $(g^i)_{1\leq i\leq n}$ est également convexe et comme la somme de deux fonctions convexes est convexe, nous en déduisons que la fonction ep_S est elle-même convexe.*

Le problème de reconnaissance de plans discrets est équivalent à un problème de réalisabilité sur une fonction convexe bi-dimensionnelle de domaine de recherche $[-1,1]^2$. Si l'ensemble S ne correspond pas à un morceau de plan discret naïf alors l'espace des solutions du problème de réalisabilité est vide. Sinon, l'espace des solutions correspond à un polygone convexe non-vide inclus dans $[-1,1]^2$.

4.3 Résoudre le Problème de Réalisabilité

4.3.1 Introduction Générale

Remarquons tout d'abord que nous ne traitons pas un problème d'optimisation mais un problème de réalisabilité. Ceci signifie que nous ne souhaitons pas déterminer d'un point u

dans le domaine de recherche tel que la valeur $ep_S(u)$ est minimum mais nous recherchons un point u tel que $ep_S(u)$ est strictement inférieure à 1. Cependant, certaines méthodes utilisées en optimisation convexe peuvent être adaptées pour résoudre des problèmes de réalisabilité.

De façon générale, pour optimiser une fonction convexe $f : \mathbb{R}^d \to \mathbb{R}$, nous devons déterminer le point x pour lequel la valeur $f(x)$ est minimum ou maximum. Pour cela, nous pouvons utiliser la méthode de descente du gradient. En utilisant cette méthode, nous approchons le minimum de la fonction en nous déplaçant itérativement dans la direction de plus forte pente. À l'itération i, le gradient $\nabla(x^i)$ de f au point x^i est calculé. Par définition du gradient, il existe une valeur réelle τ_i telle que $f(x^i)$ est inférieure à $f(x^i + \tau_i \nabla(x^i))$. Le point x^{i+1} de référence à l'itération suivante est alors égal à :

$$x^{i+1} = x^i + \tau_i \nabla(x^i) \tag{4.2}$$

Cependant, dans un souci d'efficacité, nous pouvons difficilement appliquer ce type d'algorithmes pour résoudre notre problème de réalisabilité. En effet, pour procéder à des calculs numériques exacts, les valeurs successives τ_i et par conséquent les coordonnées des points x^i doivent être représentées par des nombres rationnels. D'après (4.2), la taille du numérateur et du dénominateur de ces rationnels augmenteraient à chaque itération et ceci ralentirait significativement le calcul. Nous proposons donc une autre approche qui évite ce problème.

Notre approche pour résoudre le problème réalisabilité consiste à réduire itérativement le domaine de recherche jusqu'à ce que nous trouvions un point solution ou jusqu'à ce que le domaine de recherche soit vide. Cette réduction du domaine de recherche est faite grâce au calcul d'un sous-gradient de la fonction épaisseur.

4.3.2 Calcul du Sous-Gradient

Nous rappelons tout d'abord qu'un *sous-gradient* $g \in \mathbb{R}^d$ d'une fonction convexe $f : \mathbb{R}^d \to \mathbb{R}$ au point x vérifie : pour tout $x' \in \mathbb{R}^d$, $f(x') - f(x) \geq g \cdot (x' - x)$. Le sous-gradient est un vecteur qui indique la direction de plus forte pente de la fonction f au point x. Ceci signifie que si nous savons calculer un sous-gradient de la fonction ep_S en un point donné, nous connaissons un moyen de réduire le domaine de recherche. En effet, soit g_u un sous-gradient de la fonction ep_S au point u, alors pour tout vecteur $v \in [-1, 1]^2$ satisfaisant $g_u \cdot v \geq g_u \cdot u$, nous avons $ep_S(v) \geq ep_S(u)$. Si $ep_S(u)$ est plus grand que 1 alors $ep_S(v)$ est également plus grand que 1 et nous pouvons donc retirer tous ces points v du domaine de recherche. Les points à retirer correspondent à un des demi-plans portés par la droite de vecteur normal g_u passant par le point u. Vous remarquerez que nous utilisons le terme "sous-gradient" à la place du terme "gradient" car la fonction épaisseur n'est pas dérivable en tout point.

Pour résoudre notre problème de réalisabilité dans $\mathbb{R}^2$, nous devons déterminer comment calculer un sous-gradient de la fonction convexe ep_S à un point u donné. En accord avec la définition du sous-gradient, nous devons déterminer g tel que pour tout $v \in [-1, 1]^2$ nous avons :

$$ep_S(v) - ep_S(u) \geq g \cdot (v - u) \tag{4.3}$$

D'après la définition de ep_S, nous savons que $ep_S(u)$ est égal à $\max_{1 \leq i \leq n}(N_u \cdot p^i) - \min_{1 \leq i \leq n}(N_u \cdot p^i)$. Par conséquent, il existe deux points particuliers de S qui sont associés au produit scalaire maximum et au produit scalaire minimum. Notons respectivement p^a et p^b ces deux points (comme sur la figure 4.1), nous avons $ep_S(u) = N_u \cdot (p^a - p^b)$.

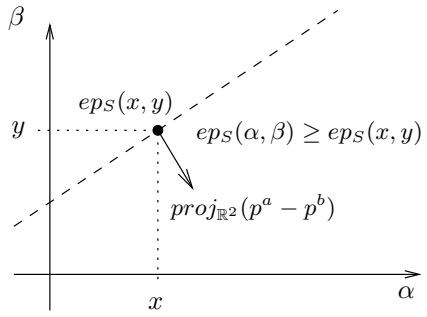


FIG. 4.1 – Un sous-gradient de ep_S

Soit T l'ensemble de deux points $\{p^a, p^b\}$. Comme T est inclus dans S, pour tout point $v \in [-1,1]^2$, l'épaisseur par rapport à l'ensemble T au point v est plus petite que l'épaisseur par rapport à l'ensemble S au point v. Nous avons :

$$\forall v \in [-1,1]^2, ep_T(v) \leq ep_S(v) \tag{4.4}$$

Comme l'ensemble T ne contient que deux points, la valeur $ep_T(v)$ est forcément égale à $|N_v \cdot (p^a - p^b)|$. De plus, nous remarquons que tout vecteur v peut être écrit comme $v + u - u$ et il s'en suit :

$$\begin{aligned} ep_T(v) &= |N_{v-u+u} \cdot (p^a - p^b)| \\ &= |N_u \cdot (p^a - p^b) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b)| \\ &= |ep_T(u) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b)| \end{aligned}$$

D'après (4.4) et par définition de la valeur absolue, nous obtenons :

$$\forall v \in \mathbb{R}^2, ep_S(v) \geq ep_T(v) \geq ep_T(u) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b) \tag{4.5}$$

Nous rappelons que nous pouvons remplacer $ep_T(u)$ par $ep_S(u)$ et nous avons donc :

$$\forall v \in \mathbb{R}^2, ep_S(v) \geq ep_S(u) + (v-u) \cdot proj_{\mathbb{R}^2}(p^a - p^b) \tag{4.6}$$

D'après (4.3), nous pouvons à présent conclure que l'expression $proj_{\mathbb{R}^2}(p^a - p^b)$ est un sous-gradient de ep_S au point u (voir la figure 4.1). Cette valeur correspond à la projection du vecteur $p^a p^b$ dans $\mathbb{R}^2$. Nous constatons donc que les deux premières coordonnées du vecteur $p^a p^b$ sont suffisantes pour déterminer localement la variation de la fonction ep_S. Lorsque nous calculons $ep_S(u)$, nous déduisons indirectement les deux points p^a et p^b. Nous calculons donc en temps constant un sous-gradient.

Remarque 4 *Nous remarquons que si les deux points p^a et p^b associés respectivement à la valeur maximum et à la valeur minimum de la fonction épaisseur en un point ont les deux mêmes premières coordonnées, alors le sous-gradient $proj_{\mathbb{R}^2}(p^a - p^b)$ correspond au vecteur nul. Dans ce cas, le sous-gradient n'indique pas la direction de plus forte pente. Cependant, si l'ensemble de points S contient au moins deux points distincts avec les deux mêmes premières composantes, alors l'ensemble S ne peut pas correspondre à un morceau de plan discret. Par conséquent, si nous calculons un sous-gradient nul, nous pouvons en déduire que l'ensemble de points ne correspond pas à un morceau de plan discret.*

4.4 Réduction du Domaine de Recherche

Nous introduisons le principe général de notre méthode. Nous appelons *coupe valide* du domaine de recherche une coupe par un demi-plan qui préserve toutes les solutions réalisables. Le principe de notre méthode est d'appliquer itérativement des coupes valides sur le domaine de recherche $[-1, 1]^2$ afin de le réduire tant qu'aucune solution réalisable est trouvée. Si le domaine est réduit à l'ensemble vide, alors le problème n'a pas de solution.

4.4.1 Coupes par le Centre de Gravité

À chaque itération, nous calculons la valeur de la fonction ep_S en un point donné u. Si $ep_S(u)$ est strictement inférieure à 1, alors u correspond à une solution de notre problème de réalisabilité et le problème est donc résolu. Dans le cas contraire, nous appliquons une coupe sur le domaine de recherche. La droite de coupe passe par le point u et elle est orthogonale au sous-gradient de la fonction épaisseur au point u. La coupe est définie par l'inégalité suivante :

$$\nabla ep_S(u) \cdot v < \nabla ep_S(u) \cdot u \tag{4.7}$$

Par définition du sous-gradient, il s'en suit que tout point v éliminé par la coupe vérifie $ep_S(v) \geq 1$. Comme chacun de ces points ne correspond pas à une solution réalisable, nous en concluons que ces coupes sont valides.

À présent, nous souhaitons nous assurer de l'efficacité de chaque coupe. En effet, nous souhaitons que chaque coupe élimine au moins un pourcentage constant du domaine de recherche courant. Pour cela, nous choisissons d'évaluer la fonction épaisseur, et donc de générer la coupe valide, en un point particulier : le *centre de gravité* $C = (C_1, C_2)$ du domaine de recherche. Nous rappelons que le centre de gravité d'un polygone est équivalent à son *barycentre*. Nous calculons ses coordonnées en temps linéaire par rapport au nombre de sommets du polygone.

La proposition suivante assure l'efficacité de chaque coupe (voir [Neu45, Grü60]).

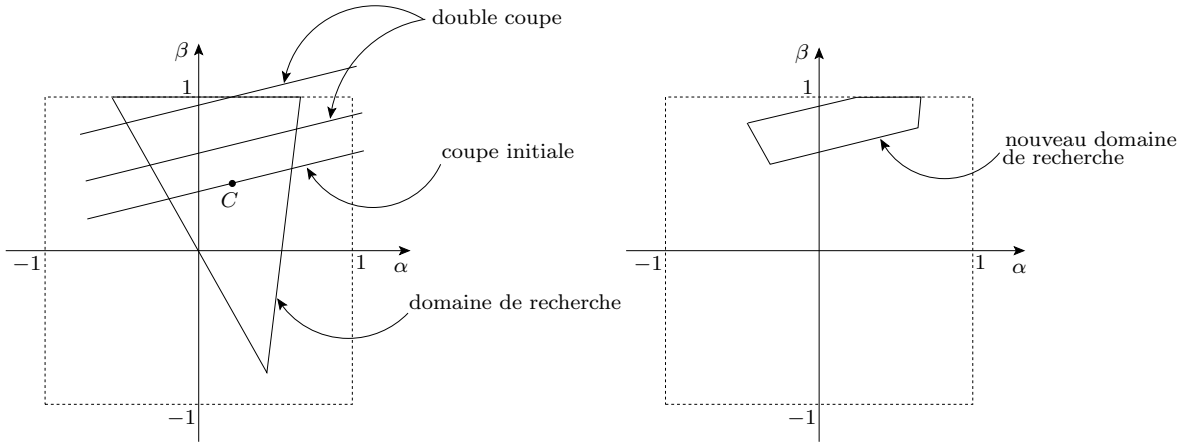
Proposition 8 *Soit K un polygone convexe dans le plan et soit C son centre de gravité. Chaque demi-plan dont la droite support passe par le centre de gravité C contient entre $4/9$ et $5/9$ de l'aire de K.*

Ceci signifie que chaque coupe passant par le centre de gravité du domaine de recherche élimine au moins $4/9$ de son aire. C'est donc au plus $5/9$ du domaine de recherche qui reste après la coupe. D'après [Neu45], nous savons que ce ratio est optimal dans le cas bi-dimensionnel. Nous pouvons remarquer qu'il est très proche du ratio $1/2$ de la dichotomie en dimension 1.

4.4.2 Amélioration des Coupes

Nous avons montré précédemment que la connaissance du sous-gradient nous permettait d'éliminer une partie du domaine de recherche. De plus, nous avons prouvé que chaque coupe passant par le centre de gravité du domaine de recherche courant éliminait au moins $4/9$ de son aire. Cependant, nous n'utilisons pas toute l'information disponible. En fait, lorsque nous calculons la valeur de $ep_S(C)$, nous déterminons deux points $p^a = (p_1^a, p_2^a, p_3^a)$ et $p^b = (p_1^b, p_2^b, p_3^b)$ de S tels que $ep_S(C) = |N_u \cdot (p^a - p^b)|$. Notons T l'ensemble de points $\{p^a, p^b\}$. Nous avons :

$$\forall v \in \mathbb{R}^2, ep_S(v) \geq ep_T(v) \tag{4.8}$$



(a) La double coupe (b) Après la double coupe

FIG. 4.2 – Double coupe du domaine de recherche

Par conséquent, nous pouvons restreindre le domaine de recherche aux vecteurs $v = (v_1, v_2)$ qui vérifient $|N_v \cdot (p^a - p^b)| < 1$. Par définition de la valeur absolue, l'inégalité $|N_v \cdot (p^a - p^b)| < 1$ est équivalente à $N_v \cdot (p^a - p^b) < 1$ et $-N_v \cdot (p^a - p^b) < 1$. Si la valeur de $ep_S(C)$ est strictement inférieure à 1, le problème est résolu. Sinon, le domaine de recherche peut être réduit en appliquant une double coupe. Soit $\nabla ep_S(C) = proj_{\mathbb{R}^2}(p^a - p^b)$ le sous-gradient de ep_S au point C, les deux coupes sont définies par les deux inégalités suivantes (voir l'exemple en figure 4.3) :

$$\nabla ep_S(C) \cdot v < 1 - (p_3^a - p_3^b) \tag{4.9}$$

$$-\nabla ep_S(C) \cdot v < 1 + (p_3^a - p_3^b) \tag{4.10}$$

Le fait de générer ces deux coupes à chaque itération accélère bien sûr l'algorithme. Néanmoins, nous n'avons aucun moyen de quantifier la portion du domaine de recherche éliminée grâce à la double coupe par rapport à la coupe simple, la complexité de la méthode est donc inchangée.

4.5 Discrétisation du Domaine de Recherche

En appliquant une coupe simple ou une double coupe sur le domaine de recherche, nous nous assurons d'éliminer au moins 4/9 de l'aire du domaine de recherche courant à chaque itération. Néanmoins, cet argument n'est pas toujours suffisant pour assurer la terminaison de l'algorithme. En effet, si le problème de réalisabilité n'admet pas de solution, l'algorithme ne déterminera jamais un domaine de recherche vide car le domaine de recherche est continu. Nous discrétisons donc le domaine de recherche afin d'assurer la terminaison de l'algorithme et pouvoir évaluer la complexité.

4.5.1 Étude de l'Espace des Solutions

Soit F l'ensemble des solutions du problème de réalisabilité. L'ensemble F est un polygone convexe et il correspond aux points u de $[-1, 1]^2$ qui satisfont $ep_S(u) < 1$. Nous prouvons par la suite que si F n'est pas vide, alors il contient un carré de côté $1/(2D^3)$ où $D-1$ correspond à la plus grande longueur d'une boite englobant l'ensemble de points

S. Cette propriété nous permet de restreindre l'espace de recherche à une grille régulière de pas de discrétisation $1/(2D^3)$ sur $[-1,1]^2$.

Nous supposons que l'ensemble de points S est contenu dans une boite englobante de taille $D-1$ et que l'origine du repère est située au centre de cette boite englobante. De ce fait, tout point p^i de S vérifie $||p^i||_\infty \leq D/2$ pour $1 \leq i \leq n$ et tout vecteur k dont les extrémités correspondent à deux points de S vérifie :

$$||k||_\infty \leq D-1 \tag{4.11}$$

Considérons l'enveloppe convexe de l'ensemble S. D'après [DR95], nous savons que la fonction épaisseur atteint sa valeur minimum pour un vecteur particulier $\overline{N} = (\overline{N}_1, \overline{N}_2, \overline{N}_3)$ tel que chacun des deux plans supports de l'ensemble S de vecteur normal $\overline{N}$ passe par au moins deux sommets de l'enveloppe convexe. Par conséquent, le vecteur $\overline{N}$ correspond au produit vectoriel de deux vecteurs dont les extrémités correspondent à ces quatre sommets (voir [PS85]). D'après (4.11), nous avons $||\overline{N}||_\infty < 2D^2$ et en particulier :

$$0 \leq \overline{N}_3 < 2D^2 \tag{4.12}$$

D'après la définition 13, si S correspond à un morceau de plan discret naïf alors $\overline{N}$ vérifie $\mu \leq \overline{N}_1 p_1^i + \overline{N}_2 p_2^i + \overline{N}_3 p_3^i \leq \mu + \overline{N}_3 - 1$ pour tout $p^i = (p_1^i, p_2^i, p_3^i)$ de S. Soit $(\overline{N}'_1, \overline{N}'_2)$ le vecteur $(\overline{N}_1/\overline{N}_3, \overline{N}_2/\overline{N}_3)$, $(\overline{N}'_1, \overline{N}'_2)$ appartient à l'espace des solutions et nous avons :

$$\forall (p_1^i, p_2^i, p_3^i) \in S,\ \mu' \leq \overline{N}'_1 p_1^i + \overline{N}'_2 p_2^i + p_3^i \leq \mu' + 1 - 1/\overline{N}_3 \tag{4.13}$$

Nous pouvons remarquer qu'en faisant légèrement varier les deux premières coordonnées du vecteur $\overline{N}$, l'ensemble S correspond toujours à la discrétisation d'un morceau de plan euclidien de vecteur normal $\overline{N}$. Soit $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2})$ le vecteur de la forme $(\overline{N}'_1 \pm \Delta, \overline{N}'_2 \pm \Delta)$ avec Δ une valeur réelle positive. Nous montrons que si $\Delta \leq 1/(4D^3)$ alors l'ensemble S correspond à un morceau de plan discret naïf de vecteur normal $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2}, 1)$ et donc le vecteur $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2})$ appartient lui aussi à l'espace des solutions. Pour cela, nous proposons de prouver la contraposée : si S ne correspond pas à un morceau de plan discret naïf de vecteur normal $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2}, 1)$ alors $\Delta > 1/(4D^3)$.

D'après (4.11) et (4.13), les inégalités suivantes sont valides :

$$\mu' - D\Delta \leq \overline{N}_{\Delta 1} p_1^i + \overline{N}_{\Delta 2} p_2^i + p_3^i \leq \mu' + 1 - 1/N_3 + D\Delta \tag{4.14}$$

Ce qui est équivalent à :

$$\mu' - D\Delta \leq \overline{N}_{\Delta 1} p_1^i + \overline{N}_{\Delta 2} p_2^i + p_3^i \leq \mu' - D\Delta + 1 + 2D\Delta - 1/\overline{N}_3 \tag{4.15}$$

D'après la définition 13 et d'après (4.15), nous déduisons que si (4.15) ne définit pas un plan discret naïf alors l'expression $2D\Delta - 1/\overline{N}_3$ est positive et donc :

$$\Delta \geq 1/(2D\overline{N}_3) \tag{4.16}$$

Il s'en suit d'après (4.12) et (4.16) que dans ce cas :

$$\Delta > 1/(4D^3) \tag{4.17}$$

Nous en déduisons que si $\Delta \leq 1/(4D^3)$ alors l'ensemble S correspond à un morceau de plan discret naïf de vecteur normal $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2}, 1)$ et donc le vecteur $(\overline{N}_{\Delta 1}, \overline{N}_{\Delta 2})$ appartient

lui aussi à l'espace des solutions. Par conséquent, nous pouvons restreindre le domaine de recherche à une grille régulière G de pas de discrétisation $1/(2D^3)$ sur $[-1,1]^2$, c'est-à-dire un carré de côté $4D^3$. En effet, pour tout ensemble de points S inclus dans une boite de longueur $D-1$ et correspondant à un morceau de plan discret naïf, il existe un carré Q de côté $1/(2D^3)$ dans $[-1,1]^2$ tel que tout point w de Q vérifie $ep_S(w) < 1$. Le choix du pas de discrétisation assure donc qu'au moins un point de la grille appartient à ce carré Q. Si aucun des points de l'échantillonnage ne correspond à une solution du problème de réalisabilité, alors l'espace des solutions est vide.

4.5.2 Amélioration du Domaine de Recherche Initial

Partir d'un domaine de recherche initial carré de côté $4D^3$ n'est pas toujours utile. Nous pouvons améliorer l'initialisation du domaine de recherche. Comme nous supposons que l'ensemble de points S est contenu dans une boite englobante de côté $D-1$, nous pouvons également supposer que certains points de S sont situés sur les bords de la boite englobante. Notons $p^l = (-D/2, p_2^l, p_3^l)$ et $p^r = (D/2, p_2^r, p_3^r)$ deux de ces points. Nous supposons que $p_2^l = p_2^r$. Ces deux points nous fournissent une contrainte supplémentaire sur l'espace des solutions. Si l'ensemble de points S correspond à un morceau de plan discret naïf de vecteur normal $N = (N_1, N_2, N_3)$, nous avons : $\gamma' \leq N \cdot (-D/2, p_2^l, p_3^l) < \gamma' + 1$ et $\gamma' \leq N \cdot (D/2, p_2^r, p_3^r) < \gamma' + 1$. Ceci implique que : $|N \cdot (D, 0, p_3^r - p_3^l)| < 1$. Nous pouvons donc en déduire la borne supérieure et inférieure suivante pour la coordonnée N_1 :

$$\frac{p_3^l - p_3^r}{D} - \frac{1}{D} < N_1 < \frac{p_3^l - p_3^r}{D} + \frac{1}{D} \tag{4.18}$$

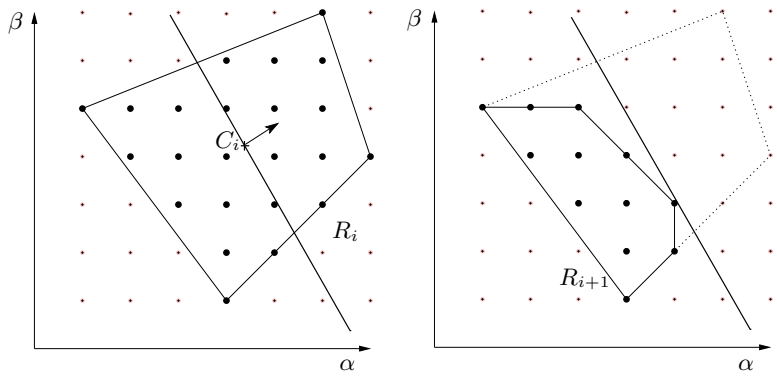
De la même manière, nous pouvons déterminer une borne supérieure et inférieure pour la coordonnée N_2 par rapport aux deux points $p^d = (p_1^d, -D/2, p_3^d)$ et $p^u = (p_1^u, D/2, p_3^u)$ de S tels que $p_1^d = p_1^u$:

$$\frac{p_3^d - p_3^u}{D} - \frac{1}{D} < N_2 < \frac{p_3^d - p_3^u}{D} + \frac{1}{D} \tag{4.19}$$

La taille des côtés du domaine de recherche est donc divisée par D. Cette étape nécessite simplement de parcourir l'ensemble de points S afin de déterminer les points extrêmes.

4.5.3 Domaine de Recherche Après Chaque Coupe

À chaque itération, nous appliquons une double coupe sur le domaine de recherche. Comme le domaine de recherche est initialisé comme un polygone convexe, il correspond toujours à un polygone convexe après chaque coupe. Néanmoins, les points d'intersection de la droite de coupe et des côtés du domaine de recherche courant correspondent à des points à coordonnées rationnelles. Par conséquent, le numérateur et le dénominateur de ces coordonnées rationnelles augmentent à chaque itération et cela a pour conséquence de ralentir l'exécution de l'algorithme. En outre, la terminaison de l'algorithme n'est pas assurée. Nous choisissons donc de décrire le domaine de recherche comme un polygone convexe à sommets portés par la grille, tout au long du déroulement de la méthode. Soit R_i le domaine de recherche au début de l'itération i. Il correspond à un polygone convexe dont les sommets sont portés par la grille. Notons à présent R'_i le domaine de recherche après la double coupe. Il correspond à un polygone convexe dont les sommets ont des coordonnées rationnelles. Soit $\overline{R'_i}$ le plus grand polygone convexe inclus dans R'_i à sommets portés par la grille du domaine de recherche. Par définition de $\overline{R'_i}$, aucun point



(a) Le domaine de recherche R_i d'aire 19.

(b) Le domaine de recherche réduit R_{i+1} d'aire 7.5 ($7.5 \leq (5/9) * 19$).

FIG. 4.3 – Coupe de R_i passant par le centre de gravité C_i

de la grille ne se situe dans $R'_i \backslash \overline{R'_i}$. Comme nous recherchons un point solution porté par la grille, nous remplaçons R_{i+1} par $\overline{R'_i}$ (voir la figure 4.3). Pour obtenir $\overline{R'_i}$ depuis R'_i, nous pouvons utiliser la méthode décrite dans la section 2.7.2 du chapitre 2 (voir aussi [CB09]) ou l'algorithme d'Harvey (voir [Har99]). Ces deux méthodes s'exécutent en $O(\log(D))$ dans le pire cas et sont efficaces en pratique.

4.6 Algorithme Résumé et Analyse de Complexité

Dans cette section, nous analysons la complexité en temps de notre algorithme de reconnaissance de plans discrets. Comme expliqué dans la section 4.4, que nous réduisions le domaine de recherche en appliquant une coupe simple ou une coupe double, cela n'a pas de conséquence sur la complexité de la méthode. Pour simplifier l'analyse de complexité, nous considérons donc que nous appliquons seulement une coupe simple sur le domaine de recherche à chaque itération, cette coupe passant par le centre de gravité du domaine de recherche courant.

4.6.1 Résumé de l'Algorithme

La toute première étape de l'algorithme consiste à initialiser le domaine de recherche R_1 comme un rectangle d'après les bornes décrites dans la section 4.5.2. Ensuite, tant que le domaine de recherche R_i contient au moins un point de la grille, l'algorithme calcule son centre de gravité C_i et il évalue la fonction épaisseur au point C_i par le biais d'un oracle. Si $ep_S(C_i)$ est strictement inférieure à 1, l'oracle retourne la valeur "vrai" et l'algorithme se termine. Ceci signifie que l'ensemble S correspond à un morceau de plan discret. Si $ep_S(C_i)$ est supérieure à 1, l'oracle calcule un sous-gradient ∇_i de la fonction épaisseur au point C_i et retourne la valeur "faux". Dans ce cas, une coupe est appliquée sur le domaine de recherche courant et l'algorithme réitère. Quand le domaine de recherche est vide, la boucle principale et donc l'algorithme se terminent. Ceci signifie que l'ensemble S ne correspond pas à un morceau de plan discret.

Durant le déroulement de l'algorithme, nous mettons à jour la liste des sommets du domaine de recherche courant R_i. Ainsi, nous pouvons connaître le nombre de ses sommets à tout moment en appelant la fonction *NombreDeSommets*. La fonction *Coupe*(R_i, C_i, ∇_i)

applique une coupe de vecteur normal ∇_i passant par le point C_i et il reconstruit le nouveau domaine de recherche R_{i+1} en utilisant la méthode de reconstruction décrite dans [CB09]. L'algorithme 15 résume notre méthode.

Algorithme 15 : Notre algorithme de reconnaissance de plans discrets

```
RECONNAISSANCE(S : ensemble de points 3D)
Valeur retournée : booléen
1  i ← 1   R_i ←InitDomaineDeRecherche(S)
2  TANT QUE NombreDeSommets(R_i) ≥ 1
3     C_i ←CentreDeGravité(R_i)
4     SI Oracle(S, C_i, ∇_i)
5        retourne VRAI
6     SINON
7        R_{i+1} ←Coupe(R_i, C_i, ∇_i)
8        i ← i + 1
9  retourne FAUX
```

4.6.2 Analyse de Complexité

Nous estimons tout d'abord le nombre d'itérations de notre méthode. Par la suite, nous analysons la complexité en temps de chaque itération.

Soit A_i l'aire du domaine de recherche R_i au début de l'itération i. L'algorithme applique une coupe valide passant par le centre de gravité C_i de R_i. Cette coupe est de la forme :

$$\nabla ep_S(C_i) \cdot v < \nabla ep_S(C_i) \cdot C_i \tag{4.20}$$

Soit R'_i le sous-ensemble de R_i correspondant aux points qui satisfont (4.20). D'après la proposition 8, l'aire de R'_i est inférieure à $(5/9)A_i$. Le domaine de recherche R_{i+1} à l'itération suivante correspond au plus grand polygone convexe à sommets portés par la grille inclus dans R'_i. Il est évident que l'aire de R_{i+1} est inférieure à l'aire de R'_i. Par conséquent, nous avons $A_{i+1} \leq (5/9)A_i$ et par extension nous avons : $A_i \leq (5/9)^i A_1$.

La figure 4.3 montre un exemple de coupe du domaine de recherche R_i passant par C_i. Nous vérifions bien que l'aire du domaine de recherche R_{i+1} est inférieure à 5/9 de l'aire du domaine R_i.

Lorsque l'aire du domaine de recherche est égale à zéro, cela signifie qu'il est réduit au polygone vide, à un simple point ou à un segment de droite. Si le domaine de recherche est réduit au polygone vide, l'ensemble de points S ne correspond pas à un morceau de plan discret naïf et l'algorithme se termine. Si le domaine de recherche est réduit à un seul point, le problème est résolu en temps linéaire puisqu'il suffit d'évaluer la fonction épaisseur en ce point restant. Dans le dernier cas, le centre de gravité correspond en fait au milieu du segment de droite. Le ratio associé à chaque coupe devient alors 1/2 et le problème est résolu en $\log_2 D$ itérations. Par conséquent, nous pouvons conclure que notre algorithme exécute $O(\log_{9/5} A_1)$ itérations dans le pire cas, où A_1 correspond à l'aire du domaine de recherche initial.

Proposition 9 *Le nombre d'itérations de notre algorithme admet une borne supérieure fonction de* A_1 *:*

$$\lfloor \log_{\frac{9}{5}} A_1 + \log_{\frac{9}{5}} 2 \rfloor + 3$$

Preuve 10 *Tant que le domaine de recherche R_i contient au moins trois points de la grille et qu'aucune solution n'a été trouvée, nous appliquons une coupe valide et l'aire du domaine de recherche R_{i+1} à l'itération suivante est inférieure à $(5/9)A_i$. Notons I_i le nombre de points de la grille contenus strictement à l'intérieur du domaine R_i. Notons B_i le nombre de points de la grille situés sur le bord de ce même domaine. Grâce au théorème de Pick (voir [Pic99] et la section 1.4.1) et comme R_i correspond à un polygone convexe à sommets portés par la grille, nous pouvons affirmer que $A_i = I_i + (B_i/2) - 1$. Par conséquent, l'inégalité suivante est toujours valide : $I_i + B_i \leq 2A_i + 2$. Pour déterminer le nombre maximum d'itérations k pour réduire le domaine de recherche à seulement deux points, nous devons donc simplement résoudre l'inéquation : $2A_k + 2 < 3$ équivalente à $2(5/9)^k A_1 < 3$. Il s'en suit que k est borné par $\log_{9/5} A_1 + \log_{9/5} 2$. Comme k est forcément une valeur entière, nous fixons cette borne supérieure à : $\lfloor \log_{9/5} A_1 + \log_{9/5} 2 \rfloor + 1$. Finalement, lorsque seuls deux points restent, nous les traitons en deux itérations au maximum. Nous remarquons que lorsque le domaine de recherche est réduit à un segment de droite alors le ratio de coupe devient $1/2$ ce qui n'augmente pas la borne supérieure pour le nombre d'itérations.*

Nous analysons à présent la complexité en temps de chaque itération. Le calcul du centre de gravité se fait en temps linéaire en fonction du nombre de sommets du domaine de recherche courant. Comme le nombre de coupes ne dépasse pas $O(\log D)$ et comme chaque coupe ajoute au plus $O(\log D)$ nouveaux sommets au domaine de recherche, le calcul du centre de gravité se fait en $O(\log^2 D)$. Par définition de la fonction épaisseur (définition 15), nous savons que l'évaluation de la fonction épaisseur en un point nécessite n produits scalaires. De plus, nous déduisons en temps constant de ce calcul un sous-gradient de la fonction épaisseur et nous connaissons donc la forme de la coupe valide à appliquer au domaine de recherche. La dernière étape consiste à reconstruire le domaine de recherche après la coupe. Pour cela, nous déterminons le plus grand polygone convexe à sommets portés par la grille inclus dans le domaine de recherche courant et dont les points satisfont l'inéquation de la coupe. Les sommets de ce polygone convexe sont situés dans une zone délimitée par la droite de coupe et deux côtés du domaine de recherche courant. Pour reconstruire le nouveau domaine de recherche, nous utilisons l'algorithme d'Harvey [Har99] ou notre méthode présentée dans la section 2.7.2 du chapitre 2 (voir aussi [CB09]). Ces deux approches s'exécutent en temps logarithmique par rapport aux coefficients de la normale de la droite de coupe et des droites portées par les deux côtés. Comme le vecteur normal de la droite de coupe correspond au sous-gradient, d'après (4.11) nous savons que ses coordonnées ne dépassent pas $O(D)$. De plus, en raison du choix du pas de discrétisation de la grille (voir Section 4.5), les coordonnées des deux autres vecteurs normaux ne dépassent pas $O(D^3)$. L'étape de reconstruction s'exécute donc en $O(\log D)$. En conclusion, comme $O(\log^2 D)$ et $O(\log D)$ peuvent être négligés par rapport à $O(n)$ (en considérant que $D \approx \sqrt{n}$), chaque coupe s'exécute en $O(n)$ dans le pire cas, ce qui implique que la complexité de notre méthode est $O(n \log D)$.

4.7 Algorithme Incrémental

Dans cette section, nous introduisons les motivations pour concevoir une version incrémentale de notre algorithme. Nous décrivons ensuite la méthode incrémentale et sa complexité.

4.7.1 Motivations et Description de l'Algorithme

En général, les données sur lesquelles nous travaillons correspondent à la dicrétisation d'objets réels. Ces données sont obtenues via des dispositifs numériques tels qu'un scanner 3D ou un télémètre laser. Après acquisition des données brutes, nous appliquons généralement des pré-traitements pour segmenter les données discrètes et sélectionner les sous-ensembles de points 3D qui sont susceptibles de correspondre à un morceau de plan discret. Ces pré-traitements utilisent généralement des critères géométriques au voisinage des points comme proposé dans [KBSS08] et dans [KBSS09]. Néanmoins, les méthodes utilisées pour discrétiser les objets continus ne fournissent pas des résultats exacts. En effet, la discrétisation d'un morceau de plan euclidien correspond généralement à un ensemble de points légèrement bombé. De plus, certains points en bordure de l'ensemble peuvent appartenir à une autre future facette de la polyédrisation de l'objet. Si nous lançons notre algorithme de reconnaissance de plans discrets sur un tel ensemble de points, il aura donc toutes les chances d'échouer. Dans ce cas, il est préférable d'exécuter l'algorithme de façon incrémentale. Pour cela, soit S l'ensemble de points tri-dimensionnel susceptible de correspondre à un morceau de plan discret, fourni par le pré-traitement. Soit S' un sous-ensemble de S ne contenant qu'un seul point. Nous ajoutons itérativement les autres points de S au sous-ensemble S', soit un par un, soit plusieurs à la fois, tant que S' correspond à un morceau de plan discret. Lorsque S' ne correspond plus à un morceau de plan discret, nous retirons le dernier point ajouté ou les derniers points ajoutés. Nous avons déterminé un sous-ensemble maximal de S correspondant à un morceau de plan discret et l'algorithme se termine. Une telle méthode est généralement appelée *croissance de région* et le premier point ajouté au sous-ensemble S' est le *point de départ* de la croissance.

Comme il y a de fortes chances pour que l'ensemble de point S soit un peu bombé, nous pouvons supposer qu'il est préférable de choisir comme point de départ le point au milieu de l'ensemble. Nous pouvons ensuite choisir les autres points à ajouter en utilisant les relations de voisinage entre les points (voir la section 1.1.1). Supposons que la projection des points de S sur le plan (x, y) est injective. À la première itération, nous ajoutons seulement le point de départ à S'. Ensuite, à chaque itération i, nous ajoutons à S' les 8-voisins des points ajoutés à l'itération précédente, soit un par un, soit tous à la fois.

4.7.2 Analyse de Complexité

La version incrémentale de notre méthode est très proche de la version non-incrémentale. Le pas de discrétisation du domaine de recherche est calculé de la même manière que dans la version non-incrémentale, en utilisant la taille d'une boite englobant l'ensemble de points S tout entier. Cependant, il n'est a priori pas possible d'initialiser le domaine de recherche par un rectangle comme décrit dans la section 4.5.2. En effet, nous ne connaissons pas les coordonnées des points qui seront ajoutés lors des itérations futures. Soit S' l'ensemble de points courant, nous appelons l'oracle en le centre de gravité de l'espace de recherche courant. Si l'oracle renvoie "faux", le domaine de recherche est réduit par application d'une coupe. Si l'oracle renvoie "vrai", un point ou une collection de points est ajouté au sous-ensemble S' que nous étudions, et l'oracle est rappelé en le même point de l'espace de recherche. Notons que dans ce cas, il est inutile de lancer l'oracle sur la totalité des points de l'ensemble S' courant. En effet, il est suffisant dans ce cas de ne tester que le ou les nouveaux points ajoutés juste avant l'appel à l'oracle ainsi que les deux points définissant le produit scalaire maximum et le produit scalaire minimum au dernier appel

de l'oracle. Lorsque le domaine de recherche est réduit à un polygone vide, nous retirons les derniers points ajoutés afin de sélectionner un ensemble de points qui correspond à un morceau de plan discret. La complexité de notre algorithme incrémental est donc en $O(n \log(D))$ dans le pire cas. En outre, nous remarquons que le fait d'ajouter les points un par un ou plusieurs à la fois ne modifie pas la complexité générale de la méthode.

L'algorithme 16 résume la version incrémentale de notre algorithme. En comparaison avec l'algorithme non-incrémental, nous avons besoin de deux fonctions supplémentaires : une pour l'ajout de points dans S', l'autre pour le retrait de points de S'. Remarquons que la fonction *ajoutePoints(S')* retourne une valeur booléenne pour indiquer si l'ensemble S' contient tous les points de S ou pas. De plus, à la fin de l'algorithme, l'ensemble de points S' correspond toujours à un morceau de plan discret. L'algorithme n'a donc plus aucune raison de retourner une valeur booléenne.

Algorithme 16 : Notre algorithme incrémental de reconnaissance de plans discrets

```
RECONNAISSANCE INCRÉMENTALE (S : ensemble de points 3D)
1  i ← 1    R_i ←InitialiseDomaineDeRecherche(S)
2  S' ← {pointMilieu(S)}     Plein ← FAUX
3  TANT QUE (NombreDeSommets(R_i) ≥ 1)
4     C_i ←CentreDeGravité(R_i)
5     SI (Oracle(S', C_i, ∇_i))
6        SI ( !Plein)
7           Plein ← ajoutePoints(S')
8        SINON
9           retourner
10    SINON
11       R_{i+1} ←Coupe(R_i, C_i, ∇_i)
12       i ← i + 1
13 RetirePoints(S')
```

4.8 Résultats Expérimentaux et Améliorations Pratiques

4.8.1 Résultats Expérimentaux

Comme l'algorithme le plus rapide que nous connaissons actuellement pour reconnaître un morceau de plan discret est l'algorithme des cordes (voir [GDRZ05]), nous allons comparer les temps d'exécution de notre méthode avec celui-ci. Pour obtenir les meilleurs temps possibles, nous testons les versions incrémentales et non-incrémentales de notre algorithme en procédant à une double coupe à chaque itération. Nous générons aléatoirement des ensembles de points tri-dimensionnels dont les coordonnées x et y sont denses dans un carré de côté $D - 1$. Nous nous assurons qu'ils correspondent à des morceaux de plans discrets naïfs. Pour créer des ensembles de points ne correspondant pas à des morceaux de plans discrets, nous modifions volontairement les coordonnées d'un point de l'ensemble. Nous testons les deux algorithmes avec ces ensembles de points. Comme l'algorithme des cordes calcule n produits scalaires à chaque itération tout comme notre méthode, nous pouvons comparer le nombre moyen d'itérations et les temps d'exécutions

	D	200	300	500	1000
Borne supérieure (itérations)	grille entière	62	67	72	79
	grille réduite	44	47	51	55
S = DP	Cordes (itérations)	10.15	10.22	10.91	11.35
	COBA (itérations)	8.24	8.56	9.44	10.10
	Cordes (temps)	0.514	1.155	3.429	14.267
	COBA (temps)	0.430	0.983	3.036	13.121
S = NDP	Cordes (itérations)	4.87	4.77	4.66	4.70
	COBA (itérations)	1.04	1.01	1.05	1.01
	Cordes (temps)	0.194	0.423	1.139	4.555
	COBA (temps)	0.056	0.116	0.330	1.257

TAB. 4.1 – Résultats expérimentaux quand S correspond à un plan discret (DP) ou pas (NDP), algorithmes non-incrémentaux

D	200	300	500	1000
cordes (itérations)	30.65	34.86	36.10	40.40
COBA (itérations)	16.21	18.25	19.55	21.80
cordes (temps)	0.274	0.640	1.330	5.742
COBA (temps)	0.237	0.521	1.286	5.272

TAB. 4.2 – Résultats expérimentaux quand S tout entier correspond à un plan discret, algorithmes incrémentaux

de ces deux méthodes. De plus, dans notre algorithme, comme les coordonnées des sommets du domaine de recherche ne peuvent pas toujours être représentées sur quatre octets, nous utilisons la bibliothèque gratuite GMP (GNU Multiple Precision Arithmetic Library) pour C++.

Les tableaux 4.1 et 4.2 montrent des résultats expérimentaux pour les versions incrémentales et non-incrémentales des deux algorithmes. Nous rappelons que $D-1$ correspond au plus grand côté d'une boite englobant l'ensemble de points S. Pour simplifier les tableaux, notre méthode est appelée COBA pour "Convex Optimization Based Algorithm". Grâce à la proposition 9, nous calculons la borne supérieure du nombre d'itérations de notre algorithme en fonction de D, dans le cas où nous partons du domaine de recherche réduit comme décrit dans la section 4.5.2 ou pas. Les résultats expérimentaux montrent qu'en moyenne notre algorithme effectue moins d'itérations et est plus efficace que l'algorithme des cordes lorsque l'ensemble de points est dense dans la boite englobante. De plus, notre méthode effectue environ cinq fois moins d'itérations en moyenne que la borne supérieure calculée. Nous remarquons également que notre algorithme incrémental est bien plus rapide que notre algorithme non-incrémental. Cela peut paraître surprenant mais ce phénomène s'explique de la manière suivante. Dans la version incrémentale, l'algorithme commence par appeler l'oracle pour des ensembles à faible nombre de points. Les premiers appels à l'oracle s'avèrent donc assez rapides tout en réduisant significativement l'espace de recherche par application de coupes. Par conséquent, l'oracle n'est appelé pour l'ensemble de points complet que très peu de fois, ce qui rend la méthode plus efficace en pratique.

4.8.2 Améliorations Pratiques

Pour obtenir de meilleurs temps d'exécution, nous pouvons ajouter deux améliorations pratiques à notre méthode.

Premièrement, nous remarquons que l'étape la plus lente de notre algorithme correspond à l'évaluation de la fonction épaisseur. Nous essayons donc d'accélérer cette phase de notre algorithme. En effet, cette étape consiste à calculer n produits scalaires afin de déterminer deux points p^a et p^b tels que le produit scalaire $C_i \cdot p^a$ est maximal et tel que le produit scalaire $C_i \cdot p^b$ est minimal. Ainsi, nous évaluons la fonction épaisseur et si celle-ci est supérieure à 1, nous calculons un sous-gradient pour générer les coupes. Nous montrons que, quand l'évaluation de la fonction épaisseur est supérieure à 1, il n'est pas toujours nécessaire de calculer n produits scalaires pour obtenir un sous-gradient. Soient p^c et p^d deux points de S tels que $C_i \cdot (p^c - p^d)$ est plus grand que 1. Soit ∇ le vecteur de la forme $proj_{\mathbb{R}^2}(p^c - p^d)$, ce vecteur correspond à un sous-gradient de la fonction épaisseur au point C_i. Par conséquent, les coupes de la forme $\nabla \cdot v < 1 - (p_3^c - p_3^d)$ et $-\nabla \cdot v < 1 + (p_3^c - p_3^d)$ correspondent à des coupes valides de notre problème de réalisabilité. Nous pouvons donc arrêter de calculer des produits scalaires dès que nous trouvons deux points p^c et p^d de S tels que $C_i \cdot (p^c - p^d)$ est supérieur à 1 et ensuite appliquer une double coupe valide sur le domaine de recherche. Bien sûr, ces coupes sont moins efficaces et donc le nombre d'itérations de notre méthode augmente. Cependant, le complexité en temps de notre algorithme est inchangée et les temps d'exécution sont significativement améliorés.

Deuxièmement, nous pouvons appliquer plus d'une double coupe à chaque itération en utilisant les points de S qui ont participé aux précédents calculs de sous-gradient. Soient p^{ai} et p^{bi} les deux points de S tels que le sous-gradient à l'itération i soit de la forme $proj_{\mathbb{R}^2}(p^{ai} - p^{bi})$. À l'itération k, nous pouvons appliquer en plus de la double coupe classique $k-1$ doubles "coupes croisées" en utilisant les couples de points p^{ak} et p^{bi} avec $1 \leq i < k$ ainsi que $k-1$ doubles coupes croisées par rapport aux couples de points p^{ai} et p^{bk} avec $1 \leq i < k$. Comme chaque double coupe s'exécute en $O(\log D)$ et comme nous pouvons considérer que $O(\log^2 D)$ peut être négligé par rapport à $O(n)$ (en considérant que $D \approx \sqrt{n}$), ce changement ne modifie pas la complexité de l'algorithme. Cependant, cela améliore légèrement son temps d'exécution.

Le tableau 4.3 montre nos résultats expérimentaux lorsque nous ajoutons ces deux améliorations pratiques : moins de tests durant l'oracle et les coupes croisées. Nous mettons également en évidence le pourcentage de points testés durant l'oracle en mettant en oeuvre la première amélioration. Pour mémoire, nous rappelons également les résultats expérimentaux de notre algorithme sans aucune des deux améliorations pratiques. Nous remarquons que ces modifications accélèrent l'exécution de l'algorithme. Par exemple, lorsque $D = 1000$, notre méthode s'exécute en environ 6.245 secondes avec les deux améliorations, contre 13.121 secondes en moyenne sans ces deux améliorations.

4.9 Conclusion

Nous décrivons une nouvelle approche pour la reconnaissance de plans discrets naïfs. Nous montrons comment transformer ce problème de reconnaissance tri-dimensionnel en un problème de réalisabilité sur une fonction convexe bi-dimensionnelle. Cette fonction convexe calcule la distance verticale entre deux plans parallèles supports encapsulant l'en-

	D	200	300	500	1000
Sans les améliorations	itérations	8.24	8.56	9.44	10.10
	temps	0.430	0.983	3.036	13.121
Moins de Tests durant Oracle (MTO)	itérations	18.24	20.07	22.38	25.80
	temps	0.302	0.670	1.666	7.372
MTO et coupes croisées	itérations	15.87	16.83	19.02	21.20
	temps	0.330	0.632	1.673	6.245
Points testés		21.4%	20.3%	16.4%	16.2%

Tab. 4.3 – Résultats expérimentaux de COBA avec améliorations pratiques

semble de points et dont les coordonnées du vecteur normal correspondent aux paramètres donnés à la fonction. Nous montrons comment évaluer cette fonction en un point et comment en déduire un sous-gradient en temps linéaire. Comme le domaine de recherche est planaire et discrétisé, nous appliquons des techniques géométriques simples pour réduire le domaine de recherche jusqu'à trouver une solution au problème de réalisabilité. De plus, nous choisissons de réduire le domaine en appliquant des coupes particulières passant par son centre de gravité ce qui assure un nombre logarithmique d'itérations. Cette version de notre algorithme atteint une complexité dans le pire cas en $O(n \log D)$. Nous présentons différentes améliorations pratiques qui accélèrent l'exécution de notre méthode. Les résultats expérimentaux que nous obtenons montrent que notre algorithme reconnaît un morceau de plan discret naïf de 10^6 voxels en 10 itérations linéaires en moyenne, ou en environ 6 secondes avec les améliorations. Ces résultats impliquent que notre algorithme est plus efficace en moyenne que l'algorithme des cordes lorsque l'ensemble de points à traiter est dense. Notre méthode a été conçue pour être efficace en pratique en particulier pour des ensembles de points denses dans une boite englobante. Il s'agit de la seule méthode connue des auteurs qui atteint une complexité de $O(n \log D)$ dans le pire cas et qui est efficace en pratique. Le problème de reconnaissance dans $\mathbb{Z}^d$ peut toujours être transformé en un problème de réalisabilité dans $\mathbb{Z}^{d-1}$, quelle que soit la dimension. Néanmoins, certaines étapes de notre méthode utilisent des algorithmes géométriques planaires dont l'extension en dimensions supérieures n'est pas évidente. C'est pourquoi, à ce jour, notre algorithme n'est valide que pour reconnaître des ensembles de points tri-dimensionnels. En outre, d'autres améliorations pratiques peuvent probablement être apportées pour accélérer encore l'exécution de la méthode.

Chapitre 5

Calcul Efficace d'Épaisseur dans un Réseau n-dimensionnel

Les travaux de recherche décrits dans ce chapitre ont été menés en collaboration avec Fabien FESCHET, professeur au laboratoire LAIC (Laboratoire d'Algorithmique et Image de Clermont), d'après une idée originale de Fabien FESCHET.

5.1 Introduction

La programmation linéaire en nombres entiers est un outil fondamental en optimisation ou encore en recherche opérationnelle. De plus, ce domaine de recherche est intéressant en lui-même puisque ce problème est NP-difficile dans le cas général. De nombreux travaux ont été menés pour le cas planaire (voir [Kan80, Sca81, HW76]) avant que Lenstra prouve que les problèmes de programmation linéaire en nombres entiers peuvent être résolus en temps polynomial quand la dimension est fixée (voir [Len83]). Des algorithmes de plus en plus rapides sont développés de nos jours, rendant l'utilisation de la programmation linéaire en nombres entiers utilisable même pour des dimensions élevées. L'approche de Lenstra utilise la notion d'*épaisseur dans un réseau* pour détecter les directions pour lesquelles le polyèdre solution est fin. Cette approche nécessite une définition précise de la notion de réseau. Ce problème est alors réduit en temps polynomial à une question de réalisabilité : étant donné un polyèdre P, déterminer si P contient un point entier. En résolvant ce problème, Lenstra approxime l'épaisseur du polyèdre et fournit une solution récursive en résolant des problèmes de dimension inférieure. L'approximation de l'épaisseur dans un réseau est également utilisée dans les récents algorithmes d'Eisenbrand et Rote [ER01] ainsi que dans ceux d'Eisenbrand et Laue [EL05] pour le problème de programmation linéaire à deux variables.

D'après l'approche arithmétique de Réveilles [Rev91, DR95], l'épaisseur dans un réseau est également un outil fondamental en géométrie discrète puisque cela correspond à la notion d'épaisseur pour les objets discrets [Fes06]. En outre, cela peut être utilisé pour détecter des structures linéaires dans des espaces discrets (voir [Fes08]). Un premier algorithme planaire a été proposé par Fabien FESCHET dans [Fes06] avec une interprétation géométrique. Cet algorithme a l'avantage d'être extensible au cas incrémental mais il semble difficilement adaptable en toute dimension. C'est pourquoi nous proposons dans ce chapitre une nouvelle méthode [CBF09] qui s'avère efficace en toute dimension et qui s'exécute en temps linéaire en dimension 2.

Le chapitre s'organise de la façon suivante. Nous rappelons dans la section 5.2 quelques définitions avant de présenter dans la section 5.3 l'algorithme planaire proposé par Feschet en 2006 (voir [Fes06]). Comme cette méthode n'est pas facilement adaptable à n'importe quelle dimension, nous introduisons dans la section 5.4 notre nouvelle méthode géométrique pour estimer l'épaisseur dans un réseau en toute dimension. Cette approche est nécessite au préalable le calcul d'un polytope englobant. Ce polytope est alors utilisé pour déterminer et borner l'ensemble des vecteurs candidats pour définir l'épaisseur. Dans la section 5.5, nous décrivons des méthodes géométriques pour déterminer un tel polytope. Nous nous intéressons tout d'abord au cas planaire et nous fournissons deux approches de complexité linéaire. Comme ces constructions géométriques semblent difficilement adaptables en toute dimension, nous décrivons un algorithme glouton efficace qui fonctionne quelle que soit la dimension.

5.2 Définitions

Dans cette section, nous rappelons quelques définitions et nous formalisons de manière précise le problème. Vous pourrez retrouver ces définitions dans [Bar02, ER01, Sch98].

Soit $\mathbb{Z}^{d*} = \mathbb{Z}^d \setminus \{0\}$ l'ensemble des vecteurs à coordonnées entières de $\mathbb{Z}^d$ différents du vecteur nul. Soit K un ensemble de n points de $\mathbb{Z}^d$. Nous supposons que les coordonnées de tous les points et vecteurs de cet ensemble peuvent être représentées sur $\log s$ bits.

Définition 20 *L'*épaisseur *de l'ensemble K* suivant *la direction $c \in \mathbb{Z}^{d*}$, notée $\omega_c(K)$ est définie par :*

$$\omega_c(K) = \max\left\{c^T x \;\middle|\; x \in K\right\} - \min\left\{c^T x \;\middle|\; x \in K\right\} \tag{5.1}$$

Géométriquement, si un ensemble K est d'épaisseur l suivant la direction c alors tout point entier de l'ensemble K est porté par un hyperplan de la forme $c^T x = \lambda$ où λ correspond à une valeur entière comprise entre $\min\{c^T x \mid x \in K\}$ et $\max\{c^T x \mid x \in K\}$. De manière un peu informelle, nous disons que l'ensemble K peut être recouvert par ces $\lfloor l \rfloor + 1$ hyperplans parallèles. Il est immédiat que $\omega(K) = \omega(\mathrm{conv}(K))$ où $\mathrm{conv}(K)$ représente l'enveloppe convexe de K. La figure 5.1 illustre un exemple de recouvrement d'un ensemble de points bi-dimensionnels par des droites. Nous voyons que pour le vecteur $c = (1, -2)$ l'épaisseur $\omega_c(K)$ est définie par les deux points $A = (0, 6)$ et $B = (7, 0)$ et vaut $c^T B - c^T A$ soit 19. En effet, l'ensemble de points est recouvert par 20 droites parallèles de la forme $c^T x = \lambda$ avec λ une valeur entière comprise entre -12 et 7.

Définition 21 *L'*épaisseur dans le réseau $\mathbb{Z}^d$ *de l'ensemble K, notée $\omega(K)$, correspond à l'épaisseur minimum suivant toutes les directions entières du réseau. Elle est définie de la manière suivante :*

$$\omega(K) = \min_{c \in \mathbb{Z}^{d*}} \omega_c(K) \tag{5.2}$$

Nous remarquons que l'épaisseur dans le réseau d'un ensemble de points entiers correspond forcément à une valeur entière.

Nous énonçons brièvement quelques propriétés importantes concernant l'inclusion et la translation.

Lemme 1 *Soient A et B deux ensembles de points de $\mathbb{Z}^d$ tels que $conv(A) \subset conv(B)$. Pour tout vecteur $c \in \mathbb{Z}^{d*}$, nous avons $\omega_c(A) \leq \omega_c(B)$. Par conséquent, nous obtenons $\omega(A) \leq \omega(B)$.*

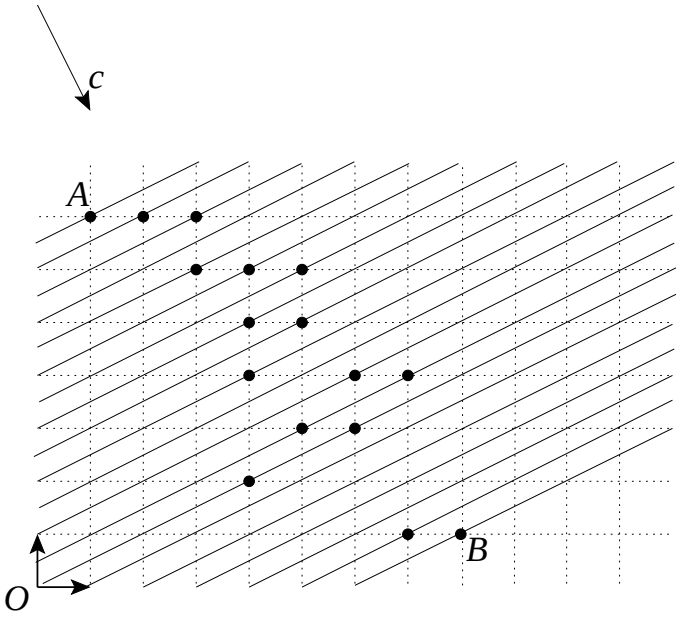


FIG. 5.1 – Épaisseur d'un ensemble de points du plan

Lemme 2 *Soit A' un ensemble de points de $\mathbb{Z}^d$ correspondant à la translation de l'ensemble A. Par définition, nous savons que pour toute direction $c \in \mathbb{Z}^{d*}$, $\omega_c(A) = \omega_c(A')$ et donc nous avons $\omega(A) = \omega(A')$. L'épaisseur dans le réseau est invariante par translation.*

Le problème que nous souhaitons résoudre est le suivant :

Problème (Épaisseur dans le réseau)
Étant donné un ensemble de points $K \subset \mathbb{Z}^d$, trouver son épaisseur dans le réseau $\omega(K)$ ainsi que les vecteurs $c \in \mathbb{Z}^d$ tels que $\omega_c(K) = \omega(K)$.

Nous savons que l'épaisseur dans le réseau d'un ensemble convexe K est obtenu pour le plus court vecteur, au sens de la norme duale, dont la boule unitaire est l'ensemble polaire de $\frac{1}{2}(K + (-K))$ [Len83]. Dans le cas général, le calcul du plus court vecteur est NP-difficile. Cependant, il est possible de calculer des solutions approximées (voir [Sch98, KS96, Rot97]) mais cela ne nous mène pas à un algorithme exact des plus simples en toute dimension.

5.3 Le Cas Planaire : Algorithme Existant

Nous rappelons dans cette partie les résultats obtenus par Feschet dans [Fes06]. Cet algorithme est lié à la notion de droiture d'une courbe discrète et plus précisément à la notion de droite discrète arithmétique (voir [Rev91]). Étant donné que cet algorithme bidimensionnel demande un polygone convexe en entrée, nous devons au préalable calculer l'enveloppe convexe H de l'ensemble K. Ce calcul d'enveloppe convexe peut se faire en $O(n)$ d'après [HKV81].

Dans [Fes06], l'idée est de remarquer que l'épaisseur dans le réseau du polygone convexe H est forcément atteinte pour deux sommets *opposés* de H (voir section 3.2).

Pour chaque droite support D, il existe au moins un sommet v de H tel que la droite D_v parallèle à D et passant par v forme avec D une bande contenant entièrement le

polygone H. Notons s un sommet de H porté par la droite D. Alors les sommets s et v forment une paire de sommets opposés (voir la figure 5.2, gauche). Notons qu'en général, une droite support n'intersecte le polygone qu'en un sommet. Cependant, il arrive qu'une telle droite intersecte le polygone le long d'un côté et dans ce cas, la droite support est qualifiée de *principale*.

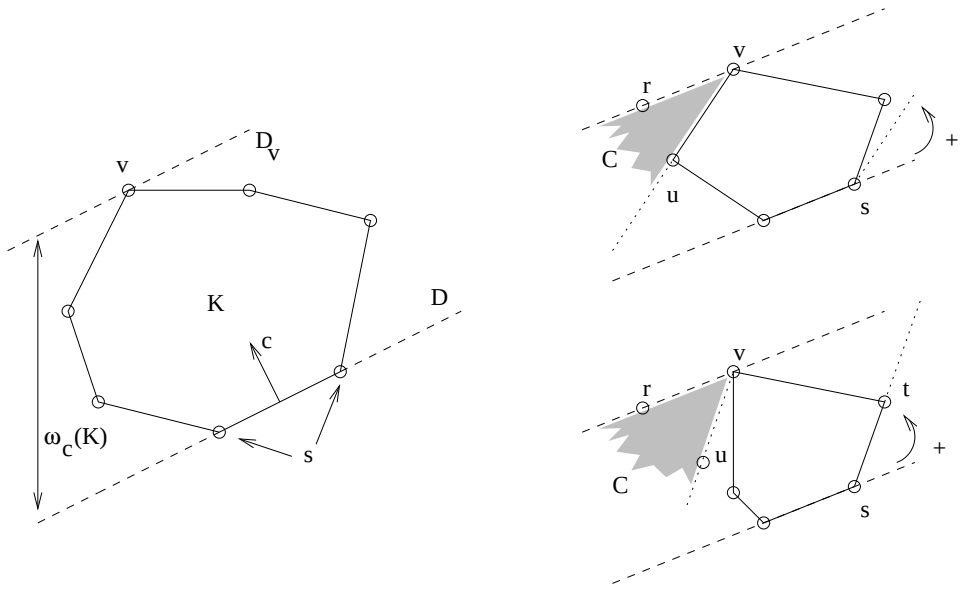


FIG. 5.2 – (gauche) Droites supports et épaisseur $\omega_c(K)$ (droite) Cône de rotation

Nous supposons à présent que les sommets de H sont orientés dans le sens inverse des aiguilles d'une montre. Soit D une droite support principale de H et D_v la droite parallèle à D telle que H est contenu dans la bande définie par D et D_v. De la même manière que dans l'algorithme bien connu du *Rotating Calipers* de Toussaint (voir [HT88]), nous pouvons faire tourner la droite support principale D autour du sommet s le plus à gauche de $D \cap H$. Nous faisons tourner la droite D_v dans le même sens afin de garder D et D_v parallèles. Nous poursuivons cette rotation jusqu'à ce que D ou D_v corresponde à une autre droite support principale et nous définissons ainsi un cône. Durant la rotation, les droites D et D_v correspondent toujours à des droites supports de H et les sommets s et v correspondent à une paire de sommets antipodaux associés à ces droites. Ainsi les sommets s et v définissent exactement $\omega_c(H)$ pour tous les vecteurs normaux c des droites inscrites dans le cône d'origine v (voir la figure 5.2 (droite)). Nous définissons le point r comme le point de la droite D_v avant la rotation tel que le segment de droite vr soit de même longueur que le côté de H opposé à la droite support D_v. De plus, nous définissons le point u de la manière suivante. Si après la rotation D_v devient une droite support principale, alors u correspond au sommet de H suivant v. Sinon, soit st le côté de H portant la nouvelle droite support principale, u correspond au point porté par la droite parallèle à st passant par v tel que la longueur du segment de droite st soit égale à la longueur du segment de droite vu.

Après avoir effectué un tour complet autour de H, nous avons construit au plus $2n$ paires de sommets antipodaux et leurs cônes associés. Le nombre de cônes est donc linéaire suivant le nombre de sommets de H. De plus, la suite des cônes forme une partition de toutes les directions possibles pour le calcul de l'épaisseur de H dans le réseau en considérant que $\omega_c(H) = \omega_{-c}(H)$. Ainsi, le calcul de l'épaisseur dans le réseau est réduit à la détermination de la plus petite valeur $\omega_c(H)$ pour chaque cône.

Comme annoncé dans la section 5.2, la plus petite valeur $\omega_c(H)$ est donnée, dans chaque cône, pour les plus courts vecteurs du cône par rapport à la norme duale. Ces plus courts vecteurs sont situés aux sommets de la voile de Klein associée au cône (voir [KS96] et la section 1.2.2). Notons que, pour avoir la possibilité de déterminer tous les vecteurs solutions, nous devons garder les répétitions dans l'enveloppe convexe. Pour calculer la voile de Klein dans chaque cône, nous pouvons adapter l'algorithme de Harvey [Har99] qui s'exécute en $O(\log s)$ opérations arithmétiques. Afin de borner la complexité de la recherche, nous pouvons nous appuyer sur le théorème général de Cook et al. (voir [CHKM92]) qui indique qu'en dimension d le nombre de sommets de la voile de Klein est borné par $O((\log s)^{d-1})$. Ce dernier résultat a également été montré par Harvey [Har99] avec un exemple bi-dimensionnel dans le pire cas à l'appui.

La complexité de cet algorithme pour le calcul de l'épaisseur dans le réseau est donc $O(n + n \log s)$.

5.4 Nouvel Algorithme

L'algorithme décrit dans la section précédente semble difficilement adaptable en toute dimension. La nouvelle méthode que nous proposons à présent est fondée sur la possibilité de borner l'ensemble des vecteurs candidats qui seront ensuite testés pour calculer l'épaisseur dans le réseau. Nous décrivons le principe de notre méthode en dimension d.

Soit $(a_j)_{1\leq j\leq d}$ une suite de d vecteurs de $\mathbb{Z}^d$ tels que pour tout $j, 1 \leq j \leq d$, il existe deux points u_j et v_j de K vérifiant $a_j = u_j - v_j$. Chacun de ces vecteurs est appelé une *corde* de l'ensemble K. Considérons un polytope englobant $\Gamma = \text{conv}((\gamma_i)_{1\leq i\leq m})$ tel que Γ contienne $\text{conv}(K)$ et tel que chacun de ses sommets γ_i, $1 \leq i \leq m$, vérifie $\gamma_i = p + \sum_{1\leq j\leq d} \alpha_{ij} a_j$. Nous présentons l'idée principale.

Si nous pouvons calculer un tel polytope englobant de façon à ce que les valeurs $|\alpha_{ij}|$, $1 \leq i \leq m$, $1 \leq j \leq d$, soient bornées par une constante α ne dépendant que de la dimension, alors nous pouvons déterminer un ensemble de vecteurs candidats C satisfaisant les propriétés suivantes : le cardinal de C est borné par une constante et l'ensemble C contient tous les vecteurs solutions de notre problème de calcul d'épaisseur dans le réseau. Par conséquent, en testant tous les vecteurs candidats, c'est-à-dire en calculant $\omega_u(K)$ pour tous les vecteurs $u \in C$ en $O(dn)$, nous déterminons l'épaisseur $\omega(K)$ dans le réseau ainsi que les vecteurs solutions associés. De plus, la constante qui borne le cardinal de C est indépendante du nombre de points de l'ensemble K et ne dépend que de la dimension d et de la constante α qui lui est associée.

Soit c un vecteur de $\mathbb{Z}^{d*}$. Par définition de l'épaisseur, nous savons que :

$$\omega_c(\Gamma) = \max_{u\in\Gamma}\{c^T u\} - \min_{u\in\Gamma}\{c^T u\}$$

Comme l'épaisseur dans un réseau est invariante par translation, nous obtenons l'inégalité suivante :

$$\omega_c(\Gamma) \leq 2 \max_{1\leq i\leq m} \{|c^T \sum_{1\leq j\leq d} \alpha_{ij} a_j|\}$$

Nous savons de plus que les valeurs $|\alpha_{ij}|, 1 \leq i \leq m, 1 \leq j \leq d$, sont bornées par une constante α, ce qui nous donne :

$$\omega_c(\Gamma) \leq 2\alpha d \sum_{1 \leq j \leq d} |c^T a_j|$$

Notons A la matrice dont les lignes correspondent aux $a_j^T, 1 \leq j \leq d$, nous pouvons réécrire l'expression précédente de la façon suivante :

$$\omega_c(\Gamma) \leq 2\alpha d ||A^T c||_\infty \tag{5.3}$$

Comme conv(K) est incluse dans conv(Γ), nous savons immédiatement que :

$$\omega(K) \leq \omega(\Gamma) \leq \omega_c(\Gamma)$$

Donc, pour un vecteur c fixé, nous obtenons une borne supérieure pour $\omega(K)$. Pour résoudre notre problème, il est suffisant de ne tester que les vecteurs u de $\mathbb{Z}^{d*}$ vérifiant :

$$\omega_u(K) \leq 2\alpha d ||A^T c||_\infty$$

Essayons à présent de déterminer une borne inférieure pour le terme $\omega_u(K)$. Pour chaque j, $1 \leq j \leq d$, il existe deux points u_j et v_j de K tels que $a_j = u_j - v_j$. Comme pour chaque j conv$(\{u_j, v_j\})$ est incluse dans conv(K), nous avons pour tout $u \in \mathbb{Z}^{d*}$, $\omega_u(\{u_j, v_j\}) \leq \omega_u(K)$. Par définition de l'épaisseur dans le réseau suivant une direction $u \in \mathbb{Z}^{d*}$, nous obtenons alors :

$$|u^T a_i| \leq \omega_u(K) \text{ for } 1 \leq i \leq d$$

Il s'en suit pour chaque $u \in \mathbb{Z}^{d*}$:

$$||A^T u||_\infty \leq \omega_u(K)$$

Nous pouvons donc conclure qu'il est suffisant de tester seulement les vecteurs u de $\mathbb{Z}^{d*}$ satisfaisant :

$$||A^T u||_\infty \leq 2\alpha d ||A^T c||_\infty \tag{5.4}$$

Comme le terme de droite est constant pour un vecteur c fixé, nous pouvons choisir judicieusement le vecteur c afin d'obtenir des propriétés intéressantes. L'approche la plus naturelle consiste à calculer le plus court vecteur v dans le réseau défini par la matrice A et de choisir le vecteur c de sorte que $A^T c = v$. La borne supérieure devient alors une borne indépendante de la direction c et nous obtenons :

$$\|v\|_\infty \leq \|A^T u\|_\infty \leq 2\alpha d \|v\|_\infty$$

Il s'en suit que l'ensemble des vecteurs à tester est contenu dans une boule dont le rayon est indépendant de l'ensemble de points K. Comme il y a un nombre constant de points entiers dans la boule, nous pouvons les tester tous afin d'obtenir l'épaisseur dans le réseau. Bien sûr, une approximation du plus court vecteur peut être utilisée à la place du plus court vecteur pour v. En effet, le calcul exact du plus court vecteur est un problème NP-difficile en dimension quelconque.

5.5 Calcul du Polytope Englobant

Soit K un ensemble de n points de $\mathbb{Z}^d$. Nous cherchons à déterminer une suite de d vecteurs $(a_j)_{1\leq j\leq d}$, d correspondant à la dimension, tels que pour chaque $j, 1 \leq j \leq d$, il existe deux points u_j et v_j de l'ensemble K satisfaisant $a_j = u_j - v_j$. À partir de ces d cordes, nous devons être capables de construire un polytope $\Gamma = \text{conv}((\gamma_i)_{1\leq i\leq m})$ englobant K tel que chaque $\gamma_i, 1 \leq i \leq m$, satisfasse $\gamma_i = \sum_{j=1}^{d} \alpha_{ij} a_j$. Le but implicite est d'obtenir la borne supérieure la plus petite possible pour les valeurs $|\alpha_{ij}|, 1 \leq i \leq m, 1 \leq j \leq d$, afin d'améliorer au maximum l'efficacité de notre algorithme. Nous montrons par la suite que, dans le cas bi-dimensionnel, nous pouvons calculer un tel polygone englobant en $O(n)$. Les approches correspondantes s'avèrent difficilement extensibles au cas tri-dimensionnel. Nous présentons donc ensuite une approche plus simple et efficace qui peut être utilisée en toute dimension.

5.5.1 Approches Linéaires 2D

Méthode Existante.

Nous avons trouvé dans la littérature le travail de Fleischer et al. (voir [FMR+92]) qui confirme qu'un tel polygone existe. Nous rappelons ce résultat dans le cas bi-dimensionnel :

Théorème 7 *Pour tout polygone convexe P, soit t le triangle de plus grande aire contenu dans P, le convexe P est lui-même contenu dans la dilatation de t par un facteur $9/4$ (à une translation près).*

D'après [HKV81], nous savons que l'enveloppe convexe d'un ensemble de points entiers dans le plan peut être calculée en temps linéaire. Nous pouvons donc travailler sur l'enveloppe convexe de l'ensemble K sans pour autant altérer la complexité globale de l'algorithme. De plus, nous savons d'après [DS79] qu'il existe un triangle de plus grande aire inclus dans K dont les sommets coincident avec des sommets de l'enveloppe convexe de K. Notons $T = A'B'C'$ la dilatation du triangle d'aire maximum $t = ABC$ de K par un facteur $9/4$. Le triangle T correspond à une solution pour notre problème de polygone englobant. En effet, si nous déplaçons l'origine du repère au point $A' + AB + AC$ par exemple, les sommets du triangle T sont de la forme $\alpha_{i1}AB + \alpha_{i2}AC$ avec $|\alpha_{i1}|$ et $|\alpha_{i2}|$ bornés par $5/4$ pour $1 \leq i \leq 3$ (voir la figure 5.3). Par conséquent, la première approche proposée consiste à calculer un triangle d'aire maximum de $\text{conv}(K)$ puis de considérer sa dilatation avec un facteur $9/4$. Boyce et al. proposent une méthode générale pour calculer un k-gone d'aire maximale qui s'exécute en $O(kn \log n + n \log^2 n)$ et en $O(n \log n)$ lorsque $k = 3$ (voir [BDDG82]). Dobkin et al. se sont plus particulièrement intéressés au cas bi-dimensionnel et ils montrent qu'il est possible de déterminer un triangle d'aire maximale en temps linéaire en réalisant au plus $10n$ calculs d'aires (voir [DS79]). Leur méthode consiste à se déplacer le long de l'enveloppe convexe de K et à déterminer pour chaque sommet v de l'enveloppe les deux autres sommets a et b tels que le triangle vab est d'aire maximale. Leur méthode est linéaire car les trois sommets sont toujours déplacés dans le sens inverse des aiguilles d'une montre et à aucun moment nous ne procédons à un "retour en arrière" (voir [DS79] pour l'algorithme détaillé).

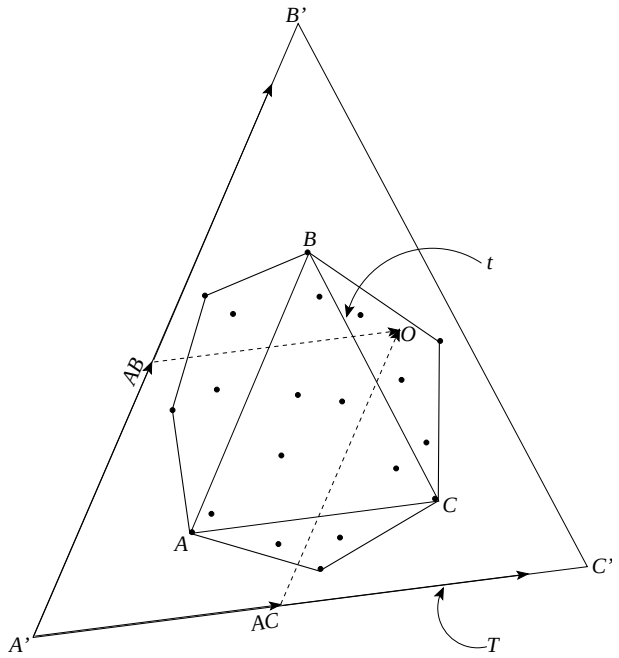


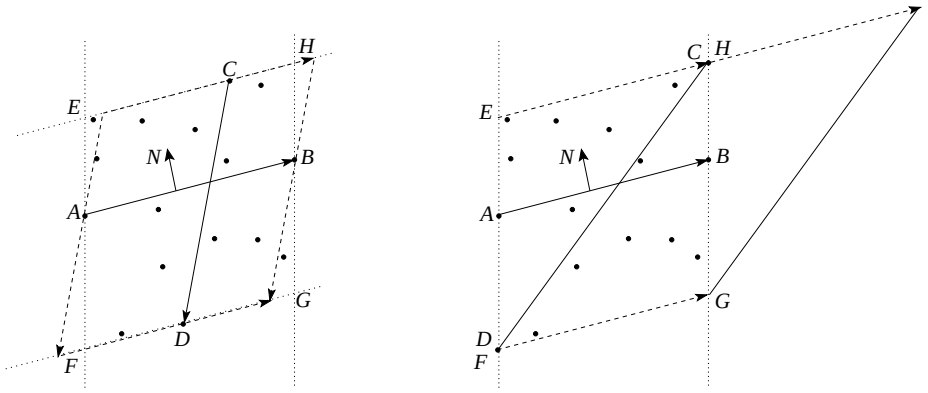
FIG. 5.3 – Triangle d'aire maximum et sa dilatation

Une Approche Plus Simple.

Nous décrivons ici une approche plus simple pour calculer un polygone englobant qui ne demande que $2n$ calculs de distances. De plus, cette méthode ne nécessite pas le pré-calcul de l'enveloppe convexe de K. Grâce à cette méthode nous calculons un parallélogramme englobant. Soient A et B les points de K situés respectivement le plus à gauche et le plus à droite par rapport à l'axe des x. Soit N le vecteur normal à AB de coordonnée y positive. Soient C et D les points de K extrémaux respectivement dans la direction N et $-N$. Nous montrons par la suite que l'ensemble K est contenu dans un parallélogramme de côté CD et $2AB$. Ce parallélogramme correspond à une solution de notre problème car ses sommets peuvent être exprimés comme $\alpha_1 CD + \alpha_2 AB$ où $|\alpha_1|$ et $|\alpha_2|$ n'excèdent pas 1 si l'origine est correctement choisie.

Notons $EFGH$ le parallélogramme borné par les deux droites verticales passant par A et B ainsi que par les deux droites de vecteur directeur AB passant par C et D. Par construction, l'ensemble K est inclus dans le parallélogramme $EFGH$, mais les côtés verticaux de ce parallélogramme ne peuvent pas être exprimés comme combinaison linéaire entière de deux cordes de K. Par conséquent, nous essayons de minimiser δ tel que le parallélogramme $EFGH$ est inclus dans le parallélogramme de côté CD et δAB. Nous montrons que δ vaut 2. En effet, les points A et B ne sont pas forcément extrémaux dans la direction du vecteur normal à CD. Dans ce cas, δ est strictement supérieur à 1 (voir Fig. 5.4.a). Le "pire cas" se produit lorsque les points C et D coincident avec des sommets opposés du parallélogramme $EFGH$, comme sur la figure 5.4.b. Dans ce cas, nous remarquons que la moitié du parallélogramme $EFGH$ est inclus dans un parallélogramme de côté AB et CD. Par conséquent, il est suffisant de doubler le côté AB du parallélogramme pour qu'il devienne assez grand.

Ces deux méthodes bi-dimensionnelles ont une complexité linéaire. Malheureusement, elles ne sont pas facilement adaptable à la dimension 3 (voir [LR03]). Par conséquent, nous décrivons dans la section suivante une approche gloutonne pour calculer un parallélépipède englobant. Cette méthode peut être étendue à toute dimension mais dans un souci de



(a) (b) Un pire cas

FIG. 5.4 – Construction des parallélogrammes

clarté, nous la décrivons en dimension 3.

5.5.2 Une Approche Gloutonne

Nous commençons par introduire la définition suivante :

Définition 22 *Soit K un ensemble de points de $\mathbb{Z}^3$. Un tétraèdre G dont les sommets appartiennent à l'ensemble K est dit* tétraèdre de croissance maximale *par rapport à l'ensemble K si pour chaque face f de G, le point de K le plus éloigné suivant la normale à f correspond à un sommet de G.*

Nous remarquons qu'un tétraèdre de croissance maximale ne correspond pas toujours à un tétraèdre de volume maximum. Un tel tétraèdre peut être exprimé en utilisant trois vecteurs a_1, a_2 et a_3 correspondant à des cordes de K. Nous montrons par la suite que nous pouvons calculer un parallélépipède Γ englobant K de sorte que chaque sommet γ_i de Γ satisfasse $\gamma_i = \sum_{1 \leq j \leq 3} \alpha_{ij} a_j$ où les valeurs $|\alpha_{ij}|$ sont bornées par 1 lorsque l'origine du repère est bien choisie.

Tout d'abord, nous calculons un tétraèdre de croissance maximale G grâce à une approche gloutonne mais efficace. De plus, cette méthode peut facilement être étendue à la dimension d contrairement aux approches linéaires définies précédemment. Notons respectivement $(f_l)_{1 \leq l \leq 4}$ et $(v_l)_{1 \leq l \leq 4}$ les faces et les sommets de G tels que le sommet v_l est opposé à la face f_l pour $1 \leq l \leq 4$. Nous initialisons G à quatre points non coplanaires de K. Notre méthode consiste à "repousser" les sommets de G parmi les points de K jusqu'à ce que G corresponde à un tétraèdre de croissance maximale. Itérativement, pour chaque face f_l de G, l'algorithme tente de faire croître le tétraèdre G en déplaçant le sommet opposé courant au point de K le plus éloigné de la face f_l. L'algorithme se termine lorsqu'aucun déplacement n'est possible en vue d'agrandir le tétraèdre. Cette méthode se termine toujours en trouvant une solution valide car à chaque étape, le volume du tétraèdre courant augmente d'au moins 1 et ce volume est borné par le volume de l'enveloppe convexe de K. L'algorithme 17 reprend le principe de la méthode.

Nous déplaçons ensuite l'origine O du repère à v_1. Cette transformation est toujours possible car l'épaisseur dans le réseau est invariante par translation. Soient $(a_j)_{1 \leq j \leq 3}$ les trois vecteurs définis par $v_{j+1} - v_1$ où les $(v_j)_{1 \leq j \leq 4}$ correspondent aux sommets d'un tétraèdre G de croissance maximale pour l'ensemble K. Nous montrons que l'ensemble

Algorithme 17 : Calcul d'un tétraèdre de croissance maximale

TÉTRAÈDRE DE CROISSANCE MAXIMALE ($K = \{k_1, k_2, ..., k_n\}$, $G = (v_1, v_2, v_3, v_4)$)
1 FAIRE
2 STABLE ← VRAI $l \leftarrow 1$
3 TANT QUE STABLE ET $l \leq 4$
4 SI $(PointLePlusLoin(K, f_l) = v_l)$
5 $l \leftarrow l + 1$
6 SINON
7 STABLE ← FAUX
8 JUSQU'À STABLE = VRAI

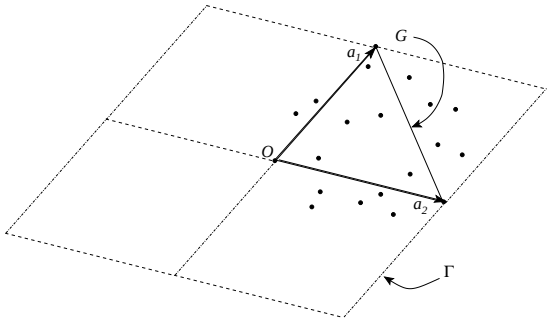


FIG. 5.5 – Exemple de parallélogramme englobant

K est contenu dans un parallélépipède Γ dont les sommets $(\gamma_i)_{1\leq i\leq 8}$ sont de la forme $\gamma_i = \sum_{1\leq j\leq 3} \alpha_{ij} a_j$ et où chaque $|\alpha_{ij}|$ est borné par 1. Comme le tétraèdre de croissance maximale que nous utilisons n'est pas plat par définition, tout sommet k_l de G peut être écrit comme $k_l = O + \sum_{1\leq j\leq 3} \delta_{lj} a_j$. Montrons à présent que nous avons $|\delta_{ij}| \leq 1$ pour $1 \leq i \leq n$ et $1 \leq j \leq d$. En effet, supposons qu'il existe un indice i et un indice j tel que l'un des coefficients $|\delta_{ij}|$ soit strictement supérieur à 1. Ceci contredirait le fait que G correspond à un tétraèdre de croissance maximale car le sommet v_{j+1} ne serait pas extrémal. Par conséquent, l'ensemble K est contenu dans un parallélépipède Γ dont les sommets $(\gamma_i)_{1\leq i\leq 8}$ sont de la forme $\gamma_i = \sum_{1\leq j\leq 3} \alpha_{ij} a_j$ et où chaque $|\alpha_{ij}|$ est borné par 1. Les figures 5.5 et 5.6 montrent respectivement des exemples dans le plan et dans l'espace : G correspond à un triangle de croissance maximale (respectivement un tétraèdre de croissance maximale) et Γ correspond à un parallélogramme englobant (respectivement à un parallélépipède englobant). Cette méthode peut être étendue à toute dimension.

5.6 Conclusion

Nous avons décrit un nouvel algorithme pour calculer l'épaisseur d'un ensemble de points K dans un réseau. En 2D, notre méthode s'exécute en temps linéaire par rapport au nombre de points de l'ensemble K, ce qui correspond à la complexité optimale. De plus, le principe de la méthode peut être étendu à des dimensions plus élevées même si les constructions intermédiaires s'avèrent plus complexes dans ce cas. Notre approche gloutonne nous permet d'appliquer notre algorithme quelle que soit la dimension puisqu'elle nous évite le calcul d'un d-simplexe inscrit de volume maximum. En effet, en 3D déjà, ce problème est en $O(n^4)$ pour le meilleur algorithme connu à ce jour (voir [LR03]). Ce

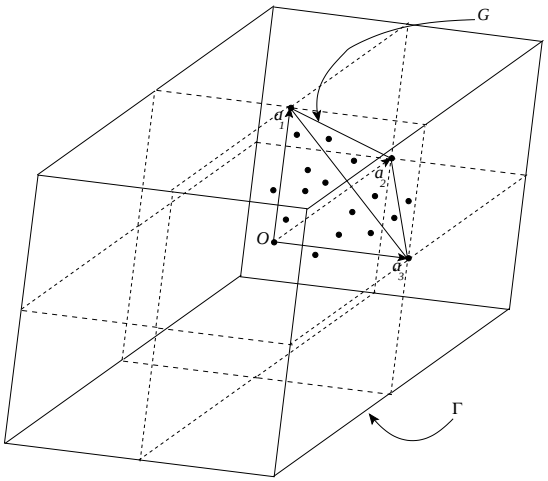


FIG. 5.6 – Exemple de parallélépipède englobant

problème est intéressant en lui-même et sera probablement le sujet de recherches futures. En effet, nous savons qu'il existe un facteur de dilatation constant d'un tel d-simplexe indépendant de K et qui garantit donc un espace de recherche le plus petit possible dans tous les cas. Il est également prévu d'étendre la définition de la linéarité donnée dans [Fes08] à une dimension arbitraire. De plus, nous prévoyons d'implémenter et de tester notre méthode afin de décrire son comportement numérique. Le fait que notre algorithme soit optimal en 2D est un point clef.

Conclusion

Les travaux menés durant ces trois années de thèse abordent différentes thématiques de la géométrie discrète et de la géométrie algorithmique. Dans l'introduction de ce mémoire, nous expliquons en détail le cheminement scientifique qui nous a amené à nous intéresser à ces problématiques. Pour conclure le manuscrit, nous proposons de reprendre les principaux résultats obtenus et d'en décrire quelques perspectives de recherche.

Approximation d'un Nombre Réel et Calcul d'Enveloppes Convexes sur une Grille Régulière

Nous avons montré que la détermination de la meilleure approximation rationnelle à dénominateur borné d'un nombre réel a est un problème qui peut être résolu par la géométrie. Pour cela, nous passons par la détermination des enveloppes convexes des points de la grille situés au-dessus et en dessous de la droite de pente a et appartenant à un domaine vertical borné. Nous avons mis en évidence le fait que les méthodes existantes pour calculer ces deux enveloppes convexes n'étaient pas complètement satisfaisantes, tant du point de vue de la complexité en temps que de l'implémentation. En effet, soit D le domaine vertical borné de la forme $\{(x, y) \in \mathbb{R}^2 | b \leq x \leq b'\}$, la méthode d'Harvey [Har99] mettant en oeuvre la théorie des nombres détermine les enveloppes convexes en $O(\log b')$, cette complexité pouvant être améliorée. Balza-Gomez et al. proposent une méthode optimale [BGMM99] utilisant la géométrie algorithmique en $O(\log(b' - b))$ mais leur algorithme n'est pas adaptatif en fonction du nombre de sommets des enveloppes calculées. La nouvelle méthode que nous proposons combine la théorie des nombres et la géométrie algorithmique. Elle conserve la complexité optimale en $O(\log(b' - b))$ et elle est la seule qui s'avère de plus adaptative. En effet, sa complexité est également linéaire en le nombre de sommets des enveloppes convexes calculées. En outre, notre algorithme est facile à implémenter ce qui constitue un de ses avantages majeurs. Notre méthode peut également être utilisée dans le cadre plus général avec un domaine borné non-vertical et une droite de référence ne passant pas par l'origine. Nous mettons d'ailleurs en oeuvre notre méthode dans le processus de reconnaissance de plans discrets que nous proposons.

Une perspective de recherche assez évidente est d'essayer d'étendre ces résultats à la dimension 3. En effet, nous souhaiterions concevoir un algorithme optimal et adaptatif pour déterminer l'enveloppe convexe des points de la grille $\mathbb{Z}^3$ situés au-dessus et en dessous d'un plan et contenus dans un domaine de la forme $\{(x, y, z) \in \mathbb{R}^3 | b \leq x \leq b' \text{ et } c \leq y \leq c'\}$. Notons qu'un tel domaine correspond à un ensemble de points dont la projection sur le plan (O, x, y) est rectangulaire. La généralisation de cet algorithme à des domaines de projection non-rectangulaire nous permettrait de plus d'implémenter notre méthode de reconnaissance d'hyperplans discrets en dimension 4. Nous avons effectué des recherches bibliographiques et il semble qu'aucune solution algorithmique n'ait été publiée à ce jour

pour résoudre ce problème de construction d'enveloppes convexes en 3D. Nous travaillons donc à l'extension de notre méthode en dimension 3. Cette étude n'est pas triviale et nous n'avons pas encore réussi à fournir une solution algorithmique viable. Cependant, nous pensons qu'une telle extension doit être possible. En dimension 2, notre méthode parvient à déterminer un nouveau sommet de l'enveloppe convexe supérieure ou inférieure grâce à un sommet de l'enveloppe convexe supérieure courante et un sommet de l'enveloppe convexe inférieure courante. À partir de ces deux sommets, notre méthode calcule en temps constant deux vecteurs entiers particuliers qui définissent un parallélogramme d'aire 1, c'est-à-dire un parallélogramme qui ne contient aucun point entier. En dimension 3, la méthode utiliserait donc de la même manière les enveloppes convexes supérieures et inférieures courantes ainsi que trois vecteurs entiers définissant un parallélépipède de volume 1. Nous ne savons actuellement pas si un tel parallélépipède existe. Notre étude se poursuit actuellement en ce sens.

Reconnaissance de Plans Discrets

Parmi les méthodes existantes pour la reconnaissance de plans discrets, rares sont celles qui combinent à la fois une complexité faible dans le pire cas et une efficacité en pratique. Nous nous sommes intéressés à la méthode de Buzer [Buz06] dont la complexité dans le pire cas est en $O(n \log^2 D)$ avec n le nombre de points de l'ensemble et $D-1$ le diamètre d'une boite englobant l'ensemble de points. Cette complexité est dite quasi-linéaire lorsque l'ensemble de points considéré est dense dans une boite englobante, c'est-à-dire lorsque D est équivalent à $\sqrt{n}$. L'algorithme de Buzer transforme le problème de reconnaissance de plans discrets en un problème de réalisabilité sur une fonction convexe bi-dimensionnelle dont le domaine de recherche est discrétisé sur $[-1, 1]^2$. Pour résoudre le problème de réalisabilité, Buzer met en place une méthode combinant l'oracle de Megiddo et une recherche dichotomique mono-dimensionnelle. Il parvient ainsi à reconnaître un morceau de plan discret d'environ 10^6 points denses dans une boite englobante en 360 itérations en moyenne. Notre travail a consisté à améliorer cette méthode en proposant des solutions algorithmiques plus efficaces pour résoudre le problème de réalisabilité. Notre approche actuelle réduit à chaque itération le domaine de recherche en appliquant une coupe passant par son centre de gravité. Grâce à une propriété du centre de gravité connue depuis les années 40, nous savons qu'au moins 4/9 de l'aire du domaine de recherche sont ainsi éliminés à chaque réduction. Ce principal changement nous a permis de ramener la complexité dans le pire cas de la méthode à $O(n \log D)$ au lieu de $O(n \log^2 D)$. De plus, notre algorithme reconnaît un morceau de plan discret de 10^6 points en 10 itérations en moyenne au lieu de 360. En ajoutant quelques améliorations pratiques, nous avons de plus pu constater l'efficacité de notre méthode comparée à l'algorithme des cordes [GDRZ05]. Nous reconnaissons alors un morceau de plan discret dense de 10^6 points en 6.245 secondes contre 14.267 secondes en moyenne pour l'algorithme des cordes.

Cet algorithme peut probablement être encore amélioré. Nous remarquons que l'étape la plus couteuse en temps d'exécution correspond à l'oracle. Nous rappelons que l'oracle calcule pour un vecteur u donné le produit scalaire $u \cdot p$ pour chaque point p de l'ensemble de départ. Les coordonnées du vecteur u correspondent à de grands entiers qui ne peuvent pas toujours être représentés sur 4 octets et pour lesquels nous utilisons donc la bibliothèque GMP. Les multiplications entre les entiers s'avèrent donc assez longues et ralentissent l'exécution de l'algorithme. Nous avons déjà proposé une amélioration pratique

qui consiste à stopper l'oracle dès que nous trouvons deux points permettant de définir une coupe valide du domaine de recherche. Cette coupe s'avère moins efficace qu'avec un oracle classique dans le sens où le nombre total d'itérations est nettement plus important. Cependant, ceci accélère globalement l'algorithme de manière très significative. Une nouvelle amélioration possible consisterait à identifier au fur et à mesure des itérations des points qui, dans le futur, n'interviendraient plus dans le calcul du sous-gradient et qui pourrait donc être supprimés. L'oracle considèrerait donc moins de points et effectuerait moins de multiplications. Nous n'avons pas encore déterminé le critère permettant de décider si un point pouvait être retiré de l'ensemble mais si un tel critère était trouvé, cela accélèrerait l'algorithme. De plus, il est toujours possible de trouver d'autres pistes pour améliorer les temps de calcul de notre méthode et cela constitue une perspective de recherche. En outre, il est envisageable de revoir le pas de discrétisation du domaine de recherche choisi car nous n'avons pas montré à ce jour s'il est optimal.

Épaisseur d'un Ensemble de Points dans le Réseau $\mathbb{Z}^d$

À partir d'une idée originale de Fabien Feschet, nous avons contribué à l'élaboration d'un algorithme pour calculer efficacement l'épaisseur d'un ensemble de points dans le réseau $\mathbb{Z}^d$. Nous rappelons que l'épaisseur $\omega_c(K)$ d'un ensemble de points K suivant un vecteur c est égale à $\max\left\{c^T x \mid x \in K\right\} - \min\left\{c^T x \mid x \in K\right\}$. De plus, l'épaisseur de l'ensemble de points K dans le réseau $\mathbb{Z}^d$ correspond à la plus petite épaisseur obtenue suivant un vecteur de $\mathbb{Z}^d$. L'algorithme proposé en 2006 par F. Feschet [Fes06] est simplement planaire et sa complexité en temps n'est pas optimale. En effet, cette méthode s'exécute en $O(n \log s)$ avec n le nombre de points et $\log s$ le nombre maximum de bits sur lesquels peuvent être représentés les vecteurs intervenant dans les calculs. La méthode que nous proposons est totalement différente de celle-ci. Elle consiste à déterminer un nombre constant de vecteurs candidats en calculant un polytope particulier englobant l'ensemble de points. L'épaisseur de l'ensemble de points suivant chaque vecteur candidat est alors calculée pour déterminer son épaisseur dans le réseau. En dimension 2, le polytope englobant et donc l'ensemble de vecteurs candidats sont déterminés en temps linéaire. La complexité totale de la méthode dans le plan est donc en $O(n)$. En dimensions supérieures, nous proposons une approche gloutonne mais efficace pour calculer le polytope.

Dans des travaux futurs, il serait intéressant de mettre en place une méthode non-gloutonne pour la détermination du polytope englobant en dimension 3 et éventuellement en dimensions supérieures. Il est également prévu d'étendre la définition de la linéarité donnée dans [Fes08] à une dimension arbitraire. De plus, nous prévoyons d'implémenter et de tester notre méthode afin de décrire son comportement numérique.

Bibliographie

[AB98] P. Alessandri and V. Berthé. Three distance theorems and combinatorics on words. *EM*, 44 :103–132, 1998.

[And79] A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Inf. Process. Lett.*, 9(5) :216–219, 1979.

[Arn98] V.I. Arnold. Higher dimensional continued fractions. *Regular and chaotic dynamics*, 3 :10–17, 1998.

[Bar02] A. Barvinok. *A Course in Convexity*, volume 54 of *Graduates Studies in Mathematics*. Amer. Math. Soc., 2002.

[BCK07] V. Brimkov, D. Coeurjolly, and R. Klette. Digital planarity-a review. *Discrete Appl. Math.*, 155(4) :468–495, 2007.

[BD07] V.E. Brimkov and S. Dantchev. Digital hyperplane recognition in arbitrary fixed dimension within an algebraic computation model. *Image Vision Comput.*, 25(10) :1631–1643, 2007.

[BDDG82] J.E. Boyce, D.P. Dobkin, R.L. Drysdale, and L.J. Guibas. Finding extremal polygons. In *STOC*, pages 282–289, 1982.

[BDH96] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4) :469–483, 1996.

[BGMM99] H. Balza-Gomez, J.-M. Moreau, and D. Michelucci. Convex hull of grid points below a line or a convex curve. In *DGCI*, pages 361–374, 1999.

[BI09] V. Berthé and L. Imbert. Diophantine approximation, ostrowski numeration and the double-base number system. *Discrete Mathematics and Theoretical Computer Science Proceedings*, 11(1) :153–172, 2009.

[Bre81] A. J. Brentjes. *Multi-dimensional continued fraction algorithms*, volume 145 of *Mathematical Centre Tracks*. Matematisch Centrum Amsterdam, 1981.

[Bur02] J. Burguet. *Surface discrètes multi-échelle, squelettisation, polyédrisation et opérations ensemblistes sur les polyèdres*. Thèse d'état, Université de Caen, Caen, France, 2002.

[Buz02] L. Buzer. *Reconnaissances des plans discrets - Simplication polygonale*. Thèse d'état, Université d'Auvergne, Clermont Ferrand, France, 2002.

[Buz03] L. Buzer. A linear incremental algorithm for naive and standard digital lines and planes recognition. *Graphical Models*, 65(1-3) :61–76, 2003.

[Buz06] L. Buzer. A composite and quasi linear time method for digital plane recognition. In *DGCI*, pages 331–342, 2006.

[Byk78] A. Bykat. Convex hull for a finite set of points in two dimensions. *IPL*, 7 :296–298, 1978.

[CB08a] E. Charrier and L. Buzer. An efficient and quasi-linear worst-case time algorithm for digital plane recognition. In *DGCI*, pages 346–357, 2008.

[CB08b] E. Charrier and L. Buzer. Matching a straight line in a two-dimensional integer domain. In *EuroCG, Collection of abstracts*, pages 173–176, 2008.

[CB08c] E. Charrier and L. Buzer. Reducing the coefficients of a two-dimensional integer linear constraint. In *IWCIA*, pages 205–216, 2008.

[CB09] E. Charrier and L. Buzer. Approximating a real number by a rational number with a limited denominator : a geometric approach. *Discrete Applied Mathematics*, 2009.

[CBF09] E. Charrier, L. Buzer, and F. Feschet. Efficient lattice width computation in arbitrary dimension. In *DGCI*, pages 46–56, 2009.

[Cha93] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10 :377–409, 1993.

[Cha96] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *GEOMETRY : Discrete & Computational Geometry*, 16 :361–368, 1996.

[CHKM92] W. Cook, M. Hartman, R. Kannan, and C. McDiarmid. On integer points in polyhedra. *Combinatorica*, 12 :27–37, 1992.

[Chr89] G. Chrystal. *Algebra : An elementary text-book Prt II.* Black, 1889.

[Chv83] V. Chvatal. *Linear Programming.* Freeman, New York, 1983.

[CK70] D. R. Chand and S. S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17 :78–86, 1970.

[Coe02] D. Coeurjolly. *Algorithmique et géométrie discrète pour la caractérisation des courbes et des surfaces.* Thèse d'état, Université Lumière Lyon 2, Laboratoire ERIC, Lyon, France, 2002.

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

[CSD+03] D. Coeurjolly, I. Sivignon, F. Dupont, F. Feschet, and J.-M. Chassery. Digital plane preimage structure. *Electronic Notes in Discrete Mathematics*, 12 :220–231, 2003.

[DA09] M. Dexet and E. Andres. A generalized preimage for the digital analytical hyperplane recognition. *Discrete Appl. Math.*, 157(3) :476–489, 2009.

[Dan98] G. Dantzig. *Linear Programming and extensions.* Princeton University Press, 1998.

[Dav92] H. Davenport. *The higher arithmetic : an introduction to the theory of numbers / by Harold Davenport.* New York : Cambridge University Press, 1992.

[dBSvKO00] M. de Berg, O. Schwarzkopf, M. van Kreveld, and M. Overmars. *Computational Geometry : Algorithms and Applications.* Springer-Verlag, 2000.

[DM80] D. P. Dobkin and J.I. Munro. Efficient uses of the past. In *FOCS*, pages 200–206, 1980.

[DR95] I. Debled-Renesson. *Étude et reconnaissance des droites et plans discrets.* Thèse d'état, Université Louis Pasteur, Strasbourg, France, 1995.

[DRR94] I. Debled-Renesson and J.-P. Reveillès. A new approach to digital planes. In *Proc. Vision Geometry III, SPIE 2356*, pages 12–21, 1994.

[DS79] D.P. Dobkin and L. Snyder. On a general method for maximizing and minimizing among certain geometric problems. In *SFCS '79 : Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 9–17, Washington, DC, USA, 1979. IEEE Computer Society.

[Dye80] M.E. Dyer. A simplified o(n log n) algorithm for the intersection of 3-polyedra. Technical report, Teesside Polytechnic, Middlesbrough UK, Department of Mathematics and Statistics, Report TMPR 80-5, 1980.

[Edd77] W. F. Eddy. A new convex hull algorithm for planar sets. *TMS*, 3 :393–403, 1977.

[Ede87] H. Edelsbrunner. *Algorithms in combinatorial geometry.* Springer-Verlag, 1987.

[EL05] F. Eisenbrand and S. Laue. A linear algorithm for integer programming in the plane. *Math. Program. Ser. A*, 102 :249–259, 2005.

[ER01] F. Eisenbrand and G. Rote. Fast 2-variable integer programming. In K. Aardal and B. Gerards, editors, *Integer Programming and Combinatorial Optimization*, volume 2081 of *LNCS*, pages 78–89. Springer-Verlag, 2001.

[Fer09] T. Fernique. Generation and recognition of digital planes using multi-dimensional continued fractions. *Pattern Recognition*, 42(10) :2229–2238, 2009.

[Fes06] F. Feschet. The exact lattice width of planar sets and minimal arithmetical thickness. In R. Reulke, U. Eckardt, B. Flach, U. Knauer, and K. Polthier, editors, *IWCIA*, volume 4040 of *Lecture Notes in Computer Science*, pages 25–33. Springer, 2006.

[Fes08] F. Feschet. The lattice width and quasi-straightness in digital spaces. In *19th International Conference on Pattern Recognition (ICPR)*, pages 1–4. IEEE, 2008.

[FMR+92] R. Fleischer, K. Mehlhorn, G. Rote, E. Welzl, and C.-K. Yap. Simultaneous inner and outer approximation of shapes. *Algorithmica*, 8(5&6) :365–389, 1992.

[FST96] J. Françon, J.-M. Schramm, and M. Tajine. Recognizing arithmetic straight lines and planes. In *DGCI'96*, pages 141–150. LNCS 1176, Springer-Verlag, 1996.

[FV98] P. Flajolet and B. Vallée. Continued fraction algorithms, functional operators and structure constants. *Theor. Comut. Sci.*, 194(1-2) :1–34, 1998.

[GÓ3] Y. Gérard. A fast and elementary algorithm for digital plane recognition. *Electronic Notes in Discrete Mathematics*, 12 :142–153, 2003.

[GDRZ05] Y. Gérard, I. Debled-Rennesson, and P. Zimmermann. An elementary digital plane recognition algorithm. *Discrete Applied Mathematics*, 151(1-3) :169–183, 2005.

[GH01] B. Gärtner and T. Herrmann. Computing the width of a point set in 3-space. In *CCCG*, pages 101–103, 2001.

[GKP94] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics : A foundation for computer science*, volume 1. Addison-Wesley Longman Publishing Co., Inc., 1994.

[Gra72] R.L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *IPL*, 1 :132–133, 1972.

[Gre79] P. J. Green. Constructing the convex hull of a set of point in the plane. *Computer Journal*, 22 :262–266, 1979.

[Grü60] B. Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes. *Pacific J. Math.*, 10 :1257–1261, 1960.

[Har99] W. Harvey. Computing two-dimensional Integer Hulls. *SIAM Journal on Computing*, 28(6) :2285–2299, 1999.

[HKV81] A. Hübler, R. Klette, and K. Voss. Determination of the convex hull of a finite set of planar points within linear time. *Elektronische Informationsverarbeitung und Kybernetik*, 17(2/3) :121–139, 1981.

[HL83] A.C. Hayes and D.G. Larman. The vertices of the knapasack polytope. *Discrete Appl. Math.*, 6 :135–138, 1983.

[HT88] M.E. Houle and G.T. Toussaint. Computing the width of a set. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(5) :761–765, 1988.

[Hun85] S.H.Y. Hung. On the straightness of digital arcs. 7(2) :203–215, 1985.

[HW76] D. Hirschberg and C.K. Wong. A polynomial-time algorithm for the knapsack problem with two variables. *J. Assoc. Comput. Mach.*, 23 :147–154, 1976.

[Jar73] R.A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *IPL*, 2 :18–21, 1973.

[Kal84] M. Kallay. The complexity of incremental convex hull algorithms in rd. *Inf. Process. Lett.*, 19(4) :197, 1984.

[Kan80] R. Kannan. A polynomial algorithm for the two variable integer programming problem. *J. Assoc. Comput. Mach.*, 27 :118–122, 1980.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4) :373–396, 1984.

[KBSS08] Y. Kenmochi, L. Buzer, A. Sugimoto, and I. Shimizu. Digital planar surface segmentation using local geometric patterns. In *DGCI*, pages 322–333, 2008.

[KBSS09] Y. Kenmochi, L. Buzer, A. Sugimoto, and I. Shimizu. Discrete plane segmentation and estimation from a point cloud using local geometric patterns. *International Journal of Autumation and Computing*, 2009. To appear.

[Keß96] C. W. Keßler. Parallel fourier-motzkin elimination. In *Euro-Pat, Vol. II*, pages 66–71, 1996.

[Kim84] C.E. Kim. Three-dimensional digital planes. *PAMI*, 6(5) :639–645, 1984.

[Kle07] F. Klein. Ausgewählte kapitel der zahlentheorie. *Teubner*, pages 17–25, 1907.

[Knu68] D. E. Knuth. *The art of computer programming / Donald E. Knuth.* Addison-Wiley, Reading, Mass., :, 1968.

[KS86] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm ? *SIAM J. Comput.*, 15(1) :287–299, 1986.

[KS91] C.E. Kim and I. Stojmenovic. On the recognition of digital planes in three-dimensional space. *PRL*, 12 :665–669, 1991.

[KS96] M. Kaib and C.-P. Schnörr. The Generalized Gauss Reduction Algorithm. *Journal of Algorithms*, 21(3) :565–578, 1996.

[KSv96] R. Klette, I. Stojmenovic, and J. Žunic. A parametrization of digital planes by least-squares fits and generalizations. *Graph. Models Image Process.*, 58(3) :295–300, 1996.

[Lac98a] G. Lachaud. Klein polygons and geometric diagrams. *Contemporary Math.*, 210 :365–372, 1998.

[Lac98b] G. Lachaud. Sails and klein polyhedra. *Contemporary Math.*, 210 :373–385, 1998.

[Len83] H.W. Lenstra. Integer Programming with a Fixed Number of Variables. *Math. Oper. Research*, 8 :535–548, 1983.

[LR03] S.I. Lyashko and B.V. Rublev. Minimal ellipsoids and maximal simplexes in 3D euclidean space. *Cybernetics and Systems Analysis*, 39(6) :831–834, 2003.

[Mat97] G.B. Mathews. On the partition of numbers. In *London Mathematical Society*, volume 28, pages 486–490, 1897.

[Meg83] N. Megiddo. Linear-time algorithms for linear programming in r3 and related problems. *SIAM J. Comput.*, 12(4) :759–776, 1983.

[Meg84] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1) :114–127, 1984.

[Mel87] A. A. Melkman. On-line construction of the convex hull of a simple polyline. *Ing. Process. Lett.*, 25(1) :11–12, 1987.

[Mor02] F. Mora. *Solving polynomial equation systems. I : The Kronecker-Duval philosophy*. Cambridge University Press, 2002.

[Mou00] J.-O. Moussafir. *Voiles et polyèdres de Klein. Géométrie algorithmes et statistiques*. Thèse d'état, Université Paris IX - DAUPHINE, Paris, France, 2000.

[Mul94] K. Mulmuley. *Computational geometry : An introduction through randomized algorithms*. Prentice-Hall, Englewood Cliffs, 1994.

[Neu45] B. H. Neumann. On an invariant of plane regions and mass distributions. *Journ. London Math. Soc.*, 20 :226–237, 1945.

[O'R98] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.

[Ost22] A. Ostrowski. Bemerkungen zur Theorie der Diophantischen Approximationen. *Abh. Math. Sem. Hamburg*, 1 :77–98,250–251, 1922.

[PBDR06] L. Provot, L. Buzer, and I. Debled-Rennesson. Recognition of blurred pieces of discrete planes. In *DGCI*, pages 65–76, 2006.

[PH77] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *CACM*, 20(1) :87–93, 1977.

[Pic99] G. Pick. Geometrisches zur zahlentheorie. *Sitzenber. Lotos*, 19 :311–319, 1899.

[Pre79] F. P. Preparata. An optimal real time algorithm for planar convex hulls. *ACM*, 22 :402–405, 1979.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry : an Introduction*. Springer-Verlag, 1985.

[Rev91] J.-P. Reveillès. *Géométrie discrète, calcul en nombres entiers et algorithmique*. Thèse d'état, Université Louis Pasteur, Strasbourg, France, 1991.

[Rev95] J.-P. Reveillès. Combinatorial pieces in digital lines and planes. In *Vision Geometry IV, SPIE 2573*, pages 23–34, 1995.

[Ros70] A. Rosenfeld. Connectivity in digital pictures. *J. ACM*, 17(1) :146–160, 1970.

[Rot97] G. Rote. Finding a shortest vector in a two-dimensional lattice modulo m. *Theoretical Computer Science*, 172(1-2) :303–308, 1997.

[Sca81] H.E. Scarf. Production sets with indivisibilities part i and part ii. *Econometrica*, 49 :1–32, 395–423, 1981.

[Sch98] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998.

[Sha94] J. Shallit. Origins of the analysis of the euclidean algorithm. *Historica Mathematica*, 21 :401–419, 1994.

[ST91] I. Stojmenovic and R. Tosic. Digitization schemes and the recognition of digital straight lines, hyperplanes and flats in arbitrary dimensions. *Vision Geometry, Contemporary Mathematics Series*, 119 :197–212, 1991.

[Ste01] S. M. Stefanov. *Separable Programming, Theory and Methods*, volume 53 of *Applied Optimization*. Kluwer Academic Publishers : Dordrecht-Boston-London, 2001.

[Tou83] G. Toussaint. Solving geometric problems with the rotating calipers. In *In Proc. IEEE MELECON '83*, pages 10–02, 1983.

[VC00] J. Vittone and J.-M. Chassery. Recognition of digital naive planes and polyhedrization. In *DGCI '00 : Proceedings of the 9th International Conference on Discrete Geometry for Computer Imagery*, pages 296–307, London, UK, 2000. Springer-Verlag.

[Vee93] P. Veelaert. On the flatness of digital hyperplanes. *JMIV*, 3 :205–221, 1993.

[Vee94] P. Veelaert. Digital planarity of rectangular surface segments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6) :647–652, 1994.

[Vui98] L. Vuillon. Combinatoire des motifs d'une suite sturmienne bidimensionnelle. *TCS*, 209 :261–285, 1998.

[Zol00] N. Y. Zolotykh. On the number of vertices in integer linear programming problems. Technical report, University of Nizhni Novgorod, 2000.

Index