École Normale Supérieure de Paris

— **D**épartement **I**nformatique —

Stage de L3
effectué au Larboratoire Lorrain de Recherche en Informatique et ses Applications

Énumeration et génération des configurations locales des plans discrets

par

Pierre Cagne

Équipe :
ADAGIo

Encadrants :
Éric Domenjoud
Damien Jamet

Juin-Août 2011

Résumé

Ce document rapporte le travail effectué dans le cadre du stage de L3 Informatique à l'École Normale Supérieure de Paris. Ce stage s'est déroulé du 1 juin au 31 août 2001 au LORIA de Nancy. Il porte sur le comptage des configurations locales des plans discrets, et plus spécialement celui des (m, n)-cubes.

Le travail s'est déroulé en plusieurs temps : un premier temps de lecture d'articles pour l'assimilation des connaissances propres au sujet et de l'état des recherches dans le domaine ; un deuxième temps d'implémentation d'algorithmes permettant le comptage expérimental des (m, n)-cubes ; enfin, un dernier temps de recherche d'algorithmes pour la génération efficace desdits (m, n)-cubes.

L'organisation de ce document suit l'évolution du travail effectué tout au long du stage.

Les différents sources, munis de leurs documentations, sont accessibles à l'adresse `http://www.eleves.ens.fr/~cagne`.

Table des matières

Introduction **1**

1 Généralités **1**
1.1 Notions 1
1.2 Résultats connus 3
1.3 Objectifs 7

2 Détermination expérimentale de $\mathrm{Card}(\mathrm{M}^0_{m,n})$ **7**
2.1 Formule d'Euler 7
2.2 Application à $\mathcal{G}^0_{m,n}$ 8
2.3 Algorithme de Bentley-Ottmann 8
2.4 Complexité et performances 13

3 Génération des éléments de $\mathrm{M}^0_{m,n}$ **13**
3.1 Idée générale 14
3.2 Graphe dual 15
3.3 Codage en parité et algorithmes 16
3.4 Complexité et implémentation 17

4 Avancées et ouvertures **18**
4.1 Réduction sur $\mathcal{G}^0_{m,n}$ 18
4.2 Cube dual 20

Introduction

La représentation en machine d'objets géométriques de $\mathbb{R}^n$ pose un problème de taille : la continuité de l'espace et la cardinalité infinie de la plupart des objets étudiés. Deux idées permettent alors l'appréhension de ces objets dans un contexte machine : premièrement, la réduction de l'espace continu à un espace **discrétisé** (typiquement $\mathbb{Z}^n$) ; deuxièmement, la **définition en compréhension** des objets plutôt qu'en extension qui permet de s'abstraire des contraintes de non finitude. La première idée est principalement une approximation gardant les propriétés de l'espace approximé (la structure de réseau remplace celle d'espace vectoriel). La deuxième idée peut sembler réductrice puisqu'elle impose que l'objet X étudié soit descriptible (typiquement par une formule) : elle ne l'est en fait que très peu étant donné que ce sont ces objets munis d'une *description naturelle* qui nous intéressent par la suite (plans, surfaces, etc.).

Cette approche est introduite pour l'étude géométrique dans $\mathbb{R}^2$ par Bresenham dans les années 60 et sera généralisée dans les années 90 par Réveillès ([RF91]) sous la forme d'**hyperplan arithmétique**. Cette approche profite naturellement de notions combinatoires telles que le comptage ou l'étude en fréquence. C'est ce point de vue combinatoire que ce document exploite dans le cas particulier des **plans arithmétiques naïfs**.

1 Généralités

1.1 Notions

On se place dans l'espace affine $\mathbb{R}^n$ euclidien muni de son produit scalaire usuel. On note $(\mathbf{e_i})_{1\leq i\leq n}$ sa base canonique.

Définition 1.1 (Hyperplans arithmétiques)**.** *Soit $n \in \mathbb{N}^*$. L'hyperplan arithmétique $\mathbf{H}(\mathbf{v}, \mu, w)$ de vecteur normal $\mathbf{v} \in \mathbb{R}^n$, de décalage $\mu \in \mathbb{R}$ et d'épaisseur $w \in \mathbb{R}_+$ est le sous-ensemble de $\mathbb{Z}^n$ défini par :*

$$\mathbf{H}(\mathbf{v}, \mu, w) = \{\mathbf{x} \in \mathbb{Z}^n, 0 \leq \langle \mathbf{x}, \mathbf{v}\rangle + \mu < w\}$$

Le terme *épaisseur* n'est pas choisi au hasard : l'hyperplan artihmétique est l'ensemble des points à coordonées entières situés entre (largement) l'hyperplan affine d'équation $\langle \mathbf{x}, \mathbf{v}\rangle + \mu = 0$ et (strictement) celui d'équation $\langle x, \mathbf{v}\rangle + \mu - w = 0$. Ces points sont donc les points à coordonnées entières à une distance moindre que $w/||\mathbf{v}||_2$ de l'hyperplan d'équation $\langle \mathbf{x}, \mathbf{v}\rangle + \mu = 0$ du côté vers lequel pointe le vecteur normal. (Voir figure 1 pour un exemple graphique.)

Parmi ces hyperplans arithmétiques s'en détachent certains appellés **hyperplans naïfs**.

Définition 1.2 (Hyperplans naïfs)**.** *Un hyperplan naïf est un hyperplan arithmétique d'épaisseur la norme infinie de son vecteur normal.*

Formellement, si $n \in \mathbb{N}^*$, l'hyperplan naïf $\mathbf{H}(\mathbf{v}, \mu)$ de vecteur normal $\mathbf{v} \in \mathbb{R}^n$ et de décalage $\mu \in \mathbb{R}$ est défini par :

$$\mathbf{H}(\mathbf{v}, \mu) = \{\mathbf{x} \in \mathbb{Z}^n, 0 \leq \langle \mathbf{x}, \mathbf{v}\rangle + \mu < ||\mathbf{v}||_\infty\}$$

Ces hyperplans naïfs sont les hyperplans arithmétiques les plus *fins* (*i.e.* de plus petite épaisseur) sans *trou*. (Le terme consacré pour *trou* est 2-tunnel. On se référera à [Jam05] pour les définitions de κ-connexité et de κ-tunnel.)

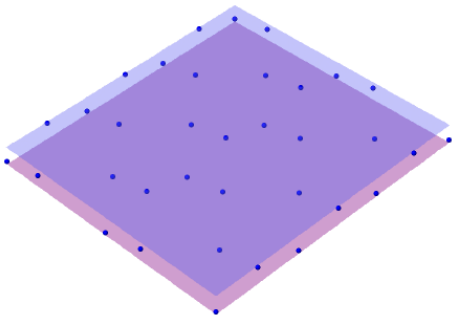


FIGURE 1 – Dans $\mathbb{R}^3$: $\mathbf{H}(\mathbf{v}, \mu, w)$ avec $\mathbf{v} = (3, 2, 6), \mu = \sqrt{2}, w = 4$

Ce document traite des plans naïfs dans $\mathbb{R}^3$. Par symétries, on peut se limiter à l'étude de ceux pour lesquels $\mathbf{v}$ est à coordonnées positives et $||\mathbf{v}||_\infty = \langle \mathbf{v}, \mathbf{e}_3 \rangle$ (le vecteur $\mathbf{v}$ est donc dans le premier octant). Par la suite, sauf mention contraire, *plan naïf* désignera un tel plan naïf. Détaillons-les quelques peu.

Comme $\mathbf{0}$ ne peut être directeur, $||\mathbf{v}||_\infty \neq 0$, et pour $\mu \in \mathbb{R}$, on a donc

$$\mathbf{P}(\mathbf{v}, \mu) = \left\{ \mathbf{x} \in \mathbb{Z}^3, 0 \leq \frac{\langle \mathbf{x}, \mathbf{v} \rangle}{||\mathbf{v}||_\infty} + \frac{\mu}{||\mathbf{v}||_\infty} < 1 \right\}$$

Sans perte de généralité, on peut donc se restreindre aux plans naïfs $\mathbf{P}(\mathbf{v}, \mu)$ avec $\mathbf{v} = (v_1, v_2, 1) \in [0, 1]^2 \times \{1\}$ et $\mu \in \mathbb{R}$. On a alors :

$$\begin{aligned} \mathbf{P}(\mathbf{v}, \mu) &= \{(x_1, x_2, x_3) \in \mathbb{Z}^3, 0 \leq x_1 v_1 + x_2 v_2 + x_3 + \mu < 1\} \\ &= \{(x, y, z) \in \mathbb{Z}^3, -z = \lfloor v_1 x + v_2 y + \mu \rfloor\} \end{aligned}$$

Ces plans naïfs sont donc finalement la discrétisation par excès des plans de $\mathbb{R}^3$ d'équation $z = -\alpha x - \beta y - \mu$, avec $\alpha, \beta \in [0, 1]$. (Voir figure 2.)

Cette *fonctionnalité en z* des plans naïfs nous montre par exemple que le plan arithmétique figure 1 n'est pas naïf : il apparaît bien que la projection du plan représenté sur $\mathbb{R}\mathbf{e_1} + \mathbb{R}\mathbf{e_2}$ n'est pas $\mathbb{Z}\mathbf{e_1} + \mathbb{Z}\mathbf{e_2}$ entier (ce n'en est qu'un sous-ensemble).

On peut alors définir ce qu'est une configuration locale rectangulaire de ces plans, comme suit.

Définition 1.3 (Motifs rectangulaire de taille $m \times n$)**.** *Soient $m, n \in \mathbb{N}$. Soit $\mathbf{P}(\mathbf{v}, \mu)$ le plan naïf de vecteur normal $\mathbf{v} \in \mathbb{R}^3$ et de décalage $\mu \in \mathbb{R}$. Le motif rectangulaire de taille $m \times n$ en $(a, b) \in \mathbb{Z}^2$ de $\mathbf{P}(\mathbf{v}, \mu)$ est le sous-ensemble suivant de ce plan naïf :*

$$\{\mathbf{x} \in [\![a, a + m - 1]\!] \times [\![b, b + n - 1]\!] \times \mathbb{Z}, 0 \leq \langle \mathbf{x}, \mathbf{v} \rangle + \mu < ||\mathbf{v}||_\infty\}$$

On notera $\mathcal{R}_{m,n,\mathbf{P}(\mathbf{v},\mu),a,b}$ le motif décrit dans cette définition. On appellera également (m, n)-*motif* un motif rectangulaire de taille $m \times n$ en une position quelconque d'un plan naïf.

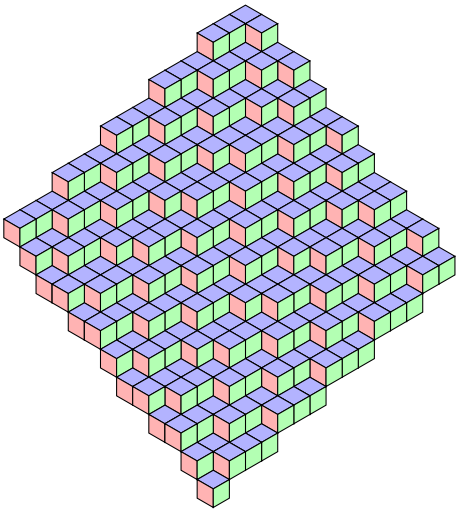


FIGURE 2 – $\mathbf{P}(\mathbf{v}, \mu)$, où $\mathbf{v} = (3, 5, 15), \mu = 3$

Visuellement, les motif rectangulaire de taille $m \times n$ d'un plan naïf $\mathbf{P}(\mathbf{v}, \mu)$ sont les images des rectangles de taille $m \times n$ de la carte $\mathbb{Z}^2$ par la fonction de définition de $\mathbf{P}(\mathbf{v}, \mu)$ (que nous avons montré bien exister).

Définition 1.4 ((m, n)-cubes)**.** *Soient $m, n \in \mathbb{N}$. Les (m, n)-cubes sont les* **classes d'équivalence** *de l'ensemble des motifs rectangulaires de tailles $m \times n$ de tous les plans naïfs pour la relation $\sim$ d'égalité modulo les translations à coordonées entières.*

Formellement, à $m, n \in \mathbb{N}$ fixé, on définit $\sim$ sur $\{\mathcal{R}_{m,n,\mathbf{P},a,b}, \mathbf{P}$ plan naïf, $(a, b) \in \mathbb{Z}^2\}$ par :

$$(\forall \mathbf{P}, \mathbf{P'} \text{ plans naïfs}, \forall a, b, a', b' \in \mathbb{Z}),$$
$$\mathcal{R}_{m,n,\mathbf{P},a,b} \sim \mathcal{R}_{m,n,\mathbf{P'},a',b'} \iff \exists \mathbf{x} \in \mathbb{Z}^3, \mathcal{R}_{m,n,\mathbf{P},a,b} = \mathbf{x} + \mathcal{R}_{m,n,\mathbf{P'},a',b'}$$

On n'hésitera pas à **identifier** un (m, n)-cube avec chacun de ses représentants. Ainsi, on pourra dire qu'**un (m, n)-cube est contenu dans un plan naïf** quand bien même il s'agit en fait d'un de ses représentants. Il s'agit finalement de faire le même abus que pour les vecteurs d'un espace affine, qui sont en fait les classes d'équivalence modulo l'égalité par translation des bi-points orientés.

1.2 Résultats connus

Le premier résultat dans le domaine est le comptage des **segments discrets** de longueur n dans $\mathbb{R}^2$. Ce sont en quelques sortes les *n-cubes* des droites du plans. Ils sont la plupart du temps définis comme les segments codés en déplacement par les facteurs de suites Sturmiennes. Ceci n'étant pas le sujet premier de ce document, on se réfèrera par exemple à [Mig91] ou à [BP93] pour des définitions précises.

Berenstein et Lavine ([BL88]), puis Mignosi ([Mig91]), montrent que le nombre de segments discrets de longueur n est

$$s(n) = 1 + \sum_{i=1}^{n-1} (n-i)\varphi(i),$$

avec φ l'indicatrice d'Euler.

Berstel et Pocchiola en donnent une très jolie preuve dans [BP93], purement géométrique, dont la démonstration du résultat suivant s'inspire fortement.

Théorème 1.1 (Nombre de $2,n$-cubes, [DJVV08])**.** *Soit $n \in \mathbb{N}\backslash\{0,1\}$. Le nombre de $2,n$-cubes est*

$$2\left(1 + \sum_{i=1}^{n} i \sum_{j=1}^{i-1} \varphi(j)\right)$$

avec φ l'indicatrice d'Euler.

Bien que la preuve ne se généralise pas entièrement aux (m,n)-cubes avec $m \in \mathbb{N}$ quelconque, elle contient des éléments importants sans restriction sur m. C'est cette partie de la preuve que nous exposons ici et qui nous servira de base pour le travail à venir.

Démonstration. Preuve partielle

Opérons tout d'abord quelques réductions.

Soit un (m,n)-motif $\mathcal{R}_{m,n,\mathbf{P}(\mathbf{v},\mu),a,b}$. $\mathbf{P}(\mathbf{v},\mu)$ est d'équation $z = -\lfloor \alpha x + \beta y + \mu \rfloor$. Alors, par définition des (m,n)-motifs, on a

$$(\forall (i,j,z) \in [\![a, a+m-1]\!] \times [\![b, b+n-1]\!] \times \mathbb{Z}),$$
$$(i,j,z) \in \mathcal{R}_{m,n,\mathbf{P}(\mathbf{v}\mu),a,b} \Leftrightarrow z = -\lfloor \alpha i + \beta j + \mu \rfloor$$

Ce qui s'écrit également,

$$(\forall (i,j,z) \in [\![0, m-1]\!] \times [\![0, n-1]\!] \times \mathbb{Z}),$$
$$(i+a, j+b, z) \in \mathcal{R}_{m,n,\mathbf{P}(\mathbf{v},\mu),a,b} \Leftrightarrow z = -\lfloor \alpha i + \beta j + (\mu + \alpha a + \beta b) \rfloor$$

Donc $\mathcal{R}_{m,n,\mathbf{P}(\mathbf{v},\mu),a,b} = (a,b,-\lfloor \mu' \rfloor) + \mathcal{R}_{m,n,\mathbf{P}(\mathbf{v},\mu'),0,0}$ où $\mu' = \alpha a + \beta b + \mu$. Ces deux motifs sont en relation et appartiennent donc à la même classe d'équivalence.

On peut ainsi se contenter de considérer les (m,n)-motifs en $(0,0)$ des plans naïfs.

De la même façon, en notant $\{\mu\} = \mu - \lfloor \mu \rfloor$ la partie fractionnaire de tout $\mu \in \mathbb{R}$, on a $\mathcal{R}_{m,n,\mathbf{P}(\mathbf{v},\mu),a,b} = (0,0,\lfloor \mu \rfloor) + \mathcal{R}_{m,n,\mathbf{P}(\mathbf{v},\{\mu\},a,b)}$. On peut alors se restreindre à $\mu \in [0,1[$.

Ces deux réductions disent en fait simplement que tout point de $\mathbb{Z}^3$ est une bonne origine pour le $\mathbb{Z}$-module affine $\mathbb{Z}^3$. Celles-ci étant faites, on peut identifier les (m,n)-motifs en $(0,0)$ et les (m,n)-cubes (en effet, translater un (m,n)-motifs d'un vecteur entier non nul fait automatiquement sortir des conditions $a=0, b=0, \mu \in [0,1[$.) On utilisera donc par abus le terme (m,n)-cube pour se référer à un (m,n)-motif réduit.

Opérons une troisième et dernière réduction. Soit un plan naïf $\mathbf{P} : z = -\lfloor \alpha x + \beta y + \mu \rfloor$. Il existe alors des perturbations sur α, β, μ telles que le plan perturbé ait le même (m,n)-motif en $(0,0)$ que le plan $\mathbf{P}$. On pose

$$\delta_\mu = \frac{1}{2} \min_{\substack{0 \le x \le m-1 \\ 0 \le y \le n-1}} (1 - \{\alpha x + \beta y + \mu\})$$

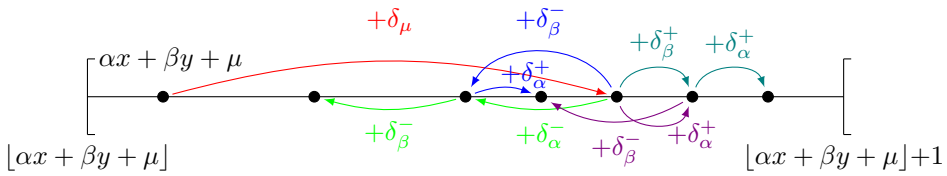
Par définition de δ_μ, on a $\lfloor \alpha x + \beta y + (\mu + \delta_\mu) \rfloor = \lfloor \alpha x + \beta y + \mu \rfloor$, $\forall (x,y) \in [\![0, m-1]\!] \times [\![0, n-1]\!]$. Comme de plus $1 - \{\xi\} > 0, (\forall \xi \in \mathbb{R})$, on a $\delta_\mu > 0$, et donc

$$\begin{aligned} &\forall (x,y) \in [\![0, m-1]\!] \times [\![0, n-1]\!] \\ &\qquad \lfloor \alpha x + \beta y + (\mu + \delta_\mu) \rfloor < \alpha x + \beta y + (\mu + \delta_\mu) < \lfloor \alpha x + \beta y + (\mu + \delta_\mu) \rfloor + 1 \end{aligned}$$

On pose alors

$$\begin{aligned} \delta_\alpha^+ &= \frac{1}{4} \min_{\substack{1 \le x \le m-1 \\ 0 \le y \le n-1}} \left(\frac{1 - \{\alpha x + \beta y + (\mu + \delta_\mu)\}}{x} \right) \\ \delta_\alpha^- &= -\frac{1}{4} \min_{\substack{1 \le x \le m-1 \\ 0 \le y \le n-1}} \left(\frac{\{\alpha x + \beta y + (\mu + \delta_\mu)\}}{x} \right) \\ \delta_\beta^+ &= \frac{1}{4} \min_{\substack{0 \le x \le m-1 \\ 1 \le y \le n-1}} \left(\frac{1 - \{\alpha x + \beta y + (\mu + \delta_\mu)\}}{y} \right) \\ \delta_\beta^- &= -\frac{1}{4} \min_{\substack{0 \le x \le m-1 \\ 1 \le y \le n-1}} \left(\frac{\{\alpha x + \beta y + (\mu + \delta_\mu)\}}{y} \right) \end{aligned}$$

Alors les plans naïfs d'équations $z = -\lfloor (\alpha + \delta_\alpha^\pm) x + (\beta + \delta_\beta^\pm) y + (\mu + \delta_\mu) \rfloor$ ont le même (m,n)-motif que $\mathbf{P}$ en $(0,0)$.



Les quatres décalages possibles.

Ceci permet donc de prendre $(\alpha, \beta) \in]0,1[^2$ plutôt que $[0,1]^2$. (On utilise bien sûr la perturbation positive pour exclure 0, et la perturbation négative pour exclure 1.)

On décide de d'abord s'intéresser au nombre de (m,n)-cubes dans les plans naïfs $\mathbf{P}(\mathbf{v},0)$, c'est-à-dire de décalage nul. Ce sont les plans naïfs d'équation sur $\mathbb{Z}^3$: $z = -\lfloor \alpha x + \beta y \rfloor$ avec $\alpha, \beta \in]0,1[$. On note $\mathrm{M}^0_{m,n}$ l'ensemble de ces (m,n)-cubes.

Si l'on définit sur l'ensemble des plans naïfs de paramètres $(\alpha,\beta,\mu) \in]0,1[^2 \times \{0\}$ la relation "$\mathbf{P}$ est en relation avec $\mathbf{P}'$ si et seulement s'ils ont le même (m,n)-motif en $(0,0)$", alors la partition $\mathfrak{P}$ que cette relation opère a exactement $\mathrm{Card}(\mathrm{M}^0_{m,n})$ parties. Il s'agit donc de détailler les *évènements* dans $]0,1[^2$ qui font changer de classe d'équivalence.

Soit $\mathcal{M} \in \mathrm{M}^0_{m,n}$ un (m,n)-cube. Alors $\mathcal{M}$ est un sous-ensemble d'un plan naïf $\mathbf{P}(\mathbf{v},0) : z = -\lfloor \alpha x + \beta y \rfloor$. Et donc

$$(\forall (x,y,z) \in \mathcal{M}),\ 0 \leq -z = \lfloor \alpha x + \beta y \rfloor < x + y \text{ car } \alpha, \beta < 1$$

Ainsi, pour (x,y) fixé, $\mathfrak{B}_{x,y} = \biguplus_{0 \leq k < x+y} \{(\alpha,\beta) \in]0,1[^2, k \leq \alpha x + \beta y < k+1\}$ est une partition de $]0,1[^2$ (qu'on appellera *partition par bandes*), et dont les éléments sont des unions d'éléments de $\mathfrak{P}$. De plus, l'ensemble de ces partitions par bandes *recouvrent* $\mathfrak{P}$, c'est à dire :

$$\forall U \in \mathfrak{P}, \exists (B_{i,j}) \in \prod_{\substack{0 \leq x \leq m-1 \\ 0 \leq y \leq n-1}} \mathfrak{B}_{x,y}, \quad U = \bigcap_{\substack{0 \leq i \leq m-1 \\ 0 \leq j \leq n-1}} B_{i,j}$$

Le plus simple est encore de voir graphiquement $\mathfrak{P}$ et son recouvrement par les partitions par bandes $\mathfrak{B}_{x,y}$ dans le carré $]0,1[^2$ (figure 3).

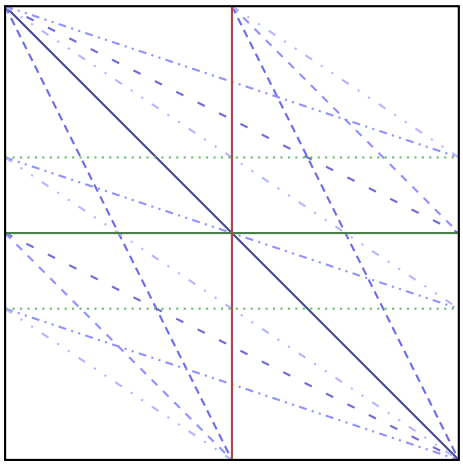


FIGURE 3 – Visualisation graphique de $\mathfrak{P}$ et des $\mathfrak{B}_{x,y}$ pour $m = 3, n = 4$

Sur la figure 3, les droites délimitant les parties d'un $\mathfrak{B}_{x,y}$ ont le même style de ligne. Les parties de $\mathfrak{P}$ sont les cellules qui apparaissent sur le graphe au final. Le terme de *partition par bandes* pour les $\mathfrak{B}_{x,y}$ prend ici tout son sens.

Le représentation graphique de $\mathfrak{P}$ est donc un graphe (noté $\mathcal{G}^0_{m,n}$ par la suite) que nous allons exploiter. En effet, afin d'obtenir $\mathrm{Card}(\mathrm{M}^0_{m,n})$, il *suffit* alors de compter le nombre de faces de ce graphe. Si ce comptage se révèle facile dans le

cas $m = 2$ (on se référera à [DJVV08] pour ce comptage, et le reste de la preuve permettant d'avoir le nombre total de $2, n$-cubes), il en est bien autrement des cas $m \in \mathbb{N}$ quelconque. C'est l'objet du travail effectué tout au long du stage. □

Bien que l'on ne l'ait pas introduit de cette manière, le graphe $\mathcal{G}^0_{m,n}$ mis en évidence est une représentation duale des objets géométriques de $\mathbb{R}^3$ étudiés. C'est en cela que la preuve s'inspire fortement de [BP93] qui réalise la même chose pour des objets géométriques de $\mathbb{R}^2$.

On verra à la section 4.2 que l'on peut pousser encore plus loin le parallèle entre [BP93] et notre problème en se plaçant carrément dans un *graphe cubique dual*.

1.3 Objectifs

Ce document propose d'exploiter au mieux le graphe $\mathcal{G}^0_{m,n}$ mis en évidence dans la démonstration du théorème 1.1. On rappelle que $\mathrm{M}^0_{m,n}$ est l'ensemble des (m, n)-cubes dont un représentant est le (m, n)-motif en $(0, 0)$ d'un plan naïf de déclage $\mu = 0$. On a alors montré que $\mathrm{M}^0_{m,n}$ est en bijection avec l'ensemble des faces de $\mathcal{G}^0_{m,n}$. Aussi, traiter les faces de ce graphe, c'est traiter les (m, n)-cubes de $\mathrm{M}^0_{m,n}$ (dans la mesure où on sait retrouver le (m, n)-cube représenté par une face).

Je me suis alors appliqué à exploiter $\mathcal{G}^0_{m,n}$ de deux manières. Premièrement, de façon purement combinatoire : pour compter le nombre de faces de $\mathcal{G}^0_{m,n}$ (*a priori* non évident), on cherche à exploiter la **formule d'Euler** en comptant expérimentalement le nombre de sommets de $\mathcal{G}^0_{m,n}$; pour cela, on implémente l'**algorithme de Bentley-Ottmann** comme exposé en section 2.3. Deuxièmement, de façon à exploiter la correspondance aux (m, n)-cubes de $\mathrm{M}^0_{m,n}$ face par face et non dans son ensemble : comme nous l'expliquerons en section 3, il peut être intéressant de générer rapidement les (m, n)-cubes ; en considérant le graphe dual (dualité au sens des graphe ici) de $\mathcal{G}^0_{m,n}$, un simple parcours sur un arbre couvrant permet alors cette génération.

2 Détermination expérimentale de $\mathrm{Card}(\mathrm{M}^0_{m,n})$

2.1 Formule d'Euler

Parcourir algorithmiquement les faces d'un graphe donné par un arrangement de segments dans le plan n'est *a priori* pas évident. Comme nous le verrons par la suite, il est bien plus simple d'en parcourir les sommets. Montrons alors comment en revenir au nombre de faces.

Soient $\mathcal{G}$ un graphe planaire fini, V l'ensemble de ses sommets, E celui de ses arrêtes et F celui de ses faces (la face externe non bornée n'est pas considéré ici comme élément de F). On note encore $v = \mathrm{Card}(V)$, $e = \mathrm{Card}(E)$ et $f = \mathrm{Card}(F)$. Alors, la formule d'Euler donne :

$$f - e + v = 1$$

Si de plus, on exploite la relation $2e = \sum_{s \in V} \deg(s)$, où deg associe à un sommet

son degré, on a :

$$f = 1 + \sum_{s \in V} \left(\frac{\deg(s)}{2} - 1 \right)$$

Ainsi, en parcourant les sommets et en comptant leur degré, on obtient facilement le nombre de faces.

2.2 Application à $\mathcal{G}^0_{m,n}$

On note $S = \{D_{i,j,k} \cap [0,1]^2, (i,j,k) \in [\![0, m-1]\!] \times [\![0, n-1]\!] \times \mathbb{N}\}$, où $D_{i,j,k}$ est la droite d'équation $ix + jy = k$ (S est donc l'union des frontières des éléments des $\mathfrak{B}_{x,y}$ de la preuve du théorème 1.1 à laquelle est ajouté la frontière du carré $]0,1[^2$ dans $\mathbb{R}^2$). On rappelle alors que l'on a défini $\mathcal{G}^0_{m,n} = (V, E)$ comme le graphe sous-jacent à S, c'est-à-dire défini par

$$\begin{aligned} V =& \{p \in \mathbb{R}^2, \quad (\exists p' \in \mathbb{R}^2, [p,p'] \in S) \vee (\exists (s,s') \in S^2, p \in s \cap s')\} \\ E =& \left\{(p,p') \in V^2, \quad \exists s \in S, [p,p'] \subseteq s\right\} \end{aligned}$$

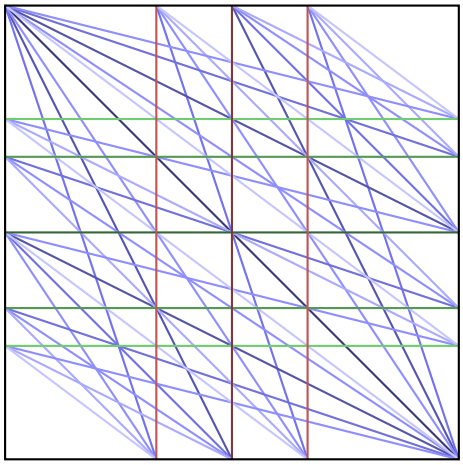


FIGURE 4 – $\mathcal{G}^0_{4,5}$

Pour appliquer la formule du 2.1, il s'agit donc de déterminer $\deg(s)$ pour chaque $s \in V$. On expose cette détermination (expérimentale) dans la partie suivante.

2.3 Algorithme de Bentley-Ottmann

On a vu précédemment que notre problème (trouver $\mathrm{Card}(\mathrm{M}^0_{m,n})$) revient à la détermination des intersections (et de leur degré) d'un arrangement de droites. Bentley et Ottmann présentent dans [BO79] une méthode de résolution de ce problème se basant sur un algorithme élaboré par Shamos et Hoey ([SH76]).

Nous présentons ici l'algorithme de Bentley-Ottmann ainsi que les points clefs de son implémentation faite au cours du stage.

Il m'a tout d'abord fallu me familiariser avec le C++, langage que je n'avais jamais pratiqué auparavant, et le concept orienté objet sous-jacent. Pour cela, j'ai d'abord codé une version *naïve*, sans traitement du cas général, de l'algorithme de Bentley-Ottmann. Cela m'a également permis de me familiariser avec les bibliothèques `Boost` (et notamment `Boost.rational`) et `Gnu MP`.

Une fois un peu plus à l'aise avec le langage, la gestion des templates et le paradigme OO dont je n'avais jusqu'alors qu'une connaissance théorique, j'ai pu implémenter une version plus générale de l'algorithme en m'appuyant notamment sur [DBCVK08], que nous présentons maintenant.

L'algorithme de Bentley-Ottmann est un algorithme dit de *sweep line* : on fait *glisser* une droite imaginaire à travers les données en assurant un certain invariant. Dans notre cas, en plaçant l'arrangement dans un repère cartésien, on *glisse* une droite verticale imaginaire de gauche à droite sur l'axe des abcisses en assurant à chaque instant que toutes les intersections strictement à gauche de la droite verticale sont traitées et que celles à droite restent à traiter.

Pour les besoins de l'algorithme, on définit l'ordre $\prec$ sur les points du plan comme l'ordre lexicographique sur les coordonnées (x, y). Étant donné un segment, on définit son *extrémité gauche* (respectivement *extrémité droite*) comme son extrémité inférieure (respectivement supérieur) pour l'ordre $\prec$. Enfin, on nommera *event* un point du plan étant une extrémité (gauche ou droite) d'un segment de l'arrangement ou une intersection de plusieurs segments de l'arrangement (le *ou* est exclusif, un event pouvant être par exemple point d'intersection et extrémité gauche et extrémité droite à la fois). Les *events* étant des points, ils seront comparés par $\prec$. On définit également pour chaque $x \in \mathbb{R}$ un ordre partiel $\prec_x$ sur les segments défini par

$$s \prec_x s' \iff \max\{(a,b) \in s, a = x\} \prec \max\{(a,b) \in s', a = x\}$$

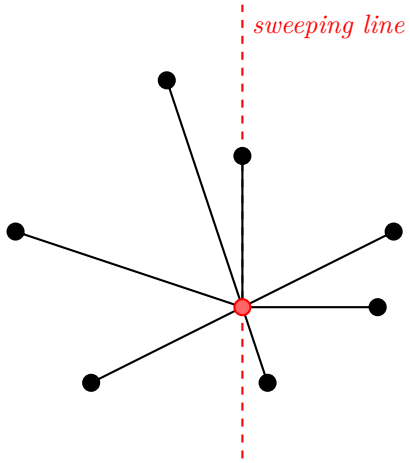
C'est-à-dire qu'en $x \in \mathbb{R}$, un segment est plus grand qu'un autre si et seulement si son point le plus haut d'abscisse x est plus haut que le point le plus haut d'abscisse x de l'autre.

L'algorithme maintient alors deux structures ordonnées de bout en bout. L'une sur les events, nommée **X-structure**. L'autre sur les segments, nommée **Y-structure**. Les segments n'étant pas totalement ordonné, on s'assurera qu'à chaque instant, la Y-structure est une chaîne maximale. Ces deux structures sont initialement vides.

L'algorithme 2.1 ne pose aucun problème quant à la X-structure. On ne fait qu'y insérer des éléments en respectant l'ordre et en supprimer l'élément minimal. L'ordre $\prec$ total sur la X-structure nous assure qu'elle reste ordonnée. Il en va autrement de la Y-structure : la relation d'ordre change au cours de l'algorithme et pour chaque position $\xi \in \mathbb{R}$ de notre droite imaginaire (*i.e.* d'équation $x = \xi$), la Y-structure est comparée par $\prec_\xi$. Il s'agit donc de vérifier que les actions effectuées en passant de la position $\xi \in \mathbb{R}$ à $\xi' \in \mathbb{R}, \xi' > \xi$ maintiennent la propriété de chaîne maximale sur la Y-structure pour $\prec_x, (\forall x \in [\xi, \xi'])$. Si $[\xi, \xi']$ ne contient pas d'event, il est clair que $\prec_\xi$ et $\prec_{\xi'}$ sont les même relations. Supposons alors qu'il existe un unique event p dans $[\xi, \xi']$ d'abscisse $x \in]\xi, \xi'[$:

- Si p est l'extrémité gauche d'un segment, alors ce segment devient comparable pour $\prec_\alpha, (\forall \alpha \in [x, \xi'])$ donc notamment pour $\prec_{\xi'}$. Il faut donc

insérer ce segment dans la Y-structure pour qu'elle garde sa propriété de chaîne maximale en ξ'.
- Si p est l'extrémité droite d'un segment, alors ce segment n'est comparable à aucun autre pour $\prec_\alpha, (\forall \alpha > x)$ donc notammant pour $\prec_{\xi'}$. Il faut donc supprimer ce segment de la Y-structure pour qu'elle garde sa propriété de chaîne maximale en ξ'.
- Si p est point d'intersection entre plusieurs segments, alors ces segments sont adjacents dans la Y-structure en ξ (sinon il y aurait un event dans $[\xi, x[$, ce qui n'est pas). Il s'agit donc juste d'inverser l'ordre de ces éléments pour que la Y-structure reste ordonnée en ξ'.



Le nombre d'events étant fini, en le notant k, on peut toujours trouver une subdivision $(x_0, \dots, x_k) \in \mathbb{R}^{k+1}$ telle que $x_0 < \dots < x_k$ et $]x_{i-1}, x_i[$ contient exactement un event pour tout $i \in [\![1, k]\!]$. Ainsi, on a décrit entièrement comment respecter la propriété de chaîne maximale de la Y-structure. D'un point de vue algorithmique, cette subdivision est produite dynamiquement (en effet, on ne connaît pas les intersections avant de lancer l'algorithme). Or, si deux segments s'intersectent, ils deviennent voisins dans la chaîne qu'est la Y-structure à un moment donné (cf. le troisième point ci-dessus). Aussi suffit-il de vérifier à chaque event si les segments devenus voisins s'intersectent, et si cela est le cas d'insérer les events (forcément à droite de la *sweeping line*, les intersections à gauche ayant déjà été traitées) dans la X-structure.

L'algorithme de Bentley-Ottmann donne une très bonne alternative à la méthode naïve (*i.e.* tester toutes les paires de segments) à condition d'implémenter de façon optimale la X-structure et la Y-structure. Ainsi, dans l'algorithme 2.1, **Q** doit être implémenté comme une file de priorité et **T** comme un arbre binaire de recherche (ou un arbre bicolore). Comme l'algorithme 2.1 requiert la recherche d'éléments dans la file de priorité, on l'implémente comme un arbre binaire de recherche ou un arbre bicolore et non classiquement comme un tas. De cette façon, toutes les insertions/suppressions/recherches sont logarithmiques en le nombre d'éléments de la structure concernée. Ainsi, en notant n le nombre de segments et k le nombre d'intersections internes de l'arrangement :
- Le tri des events en amont est en $O(n \log n)$ (n et non $n + k$, car il n'y a que les extrémités des segments à ce moment là).

Algorithme 2.1 Algorithme de Bentley-Ottmann

Paramètre(s): $\mathcal{S}$, ensemble de segments
Retour: Ensemble $\mathcal{E}$ des couples (point d'intersections, ensemble des segments s'y intersectant)

$\mathcal{E} \leftarrow \emptyset$
$\mathbf{Q} \leftarrow \emptyset$
for all $(p, q) \in \mathcal{S}$ **do**
  $\mathbf{Q} \leftarrow \mathbf{Q} \cup \{p\} \cup \{q\}$
end for
$\mathbf{T} \leftarrow \emptyset$

while $\mathbf{Q} \neq \emptyset$ **do**
  $p \leftarrow \min \mathbf{Q}$
  $\mathbf{Q} \leftarrow \mathbf{Q} \backslash \{p\}$

  $\mathcal{L}(p) \leftarrow \{s \in \mathcal{S}, \exists q \in \mathbb{R}^2, (p, q) = s\}$
  $\mathcal{R}(p) \leftarrow \{s \in \mathcal{S}, \exists q \in \mathbb{R}^2, (q, p) = s\}$
  $\mathcal{I}(p) \leftarrow \{s \in \mathcal{S}, \exists s' \in \mathcal{S}, p \in s \cap s'\}$

  if $\text{Card}(\mathcal{L}(p) \cup \mathcal{I}(p) \cup \mathcal{R}(p)) > 1$ **then**
    $\mathcal{E} \leftarrow (p, \mathcal{L}(p) \cup \mathcal{I}(p) \cup \mathcal{R}(p))$
  end if

  $\mathbf{T} \leftarrow \mathbf{T} \backslash (\mathcal{R}(p) \cup \mathcal{I}(p))$.
  $\mathbf{T} \leftarrow \mathbf{T} \cup (\mathcal{L}(p) \cup \mathcal{I}(p))$
  (* Supprimer les éléments de $\mathcal{I}(p)$ puis les réinsérer inverse leur ordre dans $\mathbf{T}$. *)

  if $\mathcal{U}(p) \cup \mathcal{I}(p) \neq \emptyset$ **then**
    $s_a \leftarrow \min\{s \in \mathbf{T}, (p, p) \prec_{x_p} s\}$
    $s_b \leftarrow \max\{s \in \mathbf{T}, s \prec_{x_p} (p, p)\}$

    if s_a défini **and** s_b défini **and** $s_a \cap s_b = \{i\}$ **then** (* Nouveaux events. *)
      if $p \prec i$ **then**
        $\mathbf{Q} \leftarrow \mathbf{Q} \cup \{i\}$
      end if
    end if

  else
    $s'_a \leftarrow \max_{\prec_{x_p}} \mathcal{U}(p) \cup \mathcal{I}(p)$
    $s_a \leftarrow \min\{s \in \mathbf{T}, (p, p) \prec_{x_p} s\}$

    if s_a défini **and** s'_a défini **and** $s_a \cap s'_a = \{i\}$ **then** (* Nouveaux events. *)
      if $p \prec i$ **then**
        $\mathbf{Q} \leftarrow \mathbf{Q} \cup \{i\}$
      end if
    end if

    $s'_b \leftarrow \min_{\prec_{x_p}} \mathcal{U}(p) \cup \mathcal{I}(p)$
    $s_b \leftarrow \max\{s \in \mathbf{T}, s \prec_{x_p} (p, p)\}$

    if s_b défini **and** s'_b défini **and** $s_b \cap s'_b = \{i\}$ **then** (* Nouveaux events. *)
      if $p \prec i$ **then**
        $\mathbf{Q} \leftarrow \mathbf{Q} \cup \{i\}$
      end if
    end if

  end if
end while
return $\mathcal{E}$

– La bloucle **while** principale est un $O((n+k)\gamma(n,k))$ avec γ la complexité des instructions intérieures à la boucle.
– Les instructions intérieurs non constantes en temps sont soit les insertions, suppressions et recherches dans la structure $\mathbf{T}$ (trivialement en $O(\log n)$), soit l'insertion de nouveaux events dans la structure $\mathbf{Q}$ (donc en $O(\log(n+k)) = O(\log n)$ puisque $k \leq n(n-1)/2$). Et donc $\gamma(n,k) = O(\log n)$.

Ainsi, la complexité temporelle finale de l'algorithme de Bentley-Ottmann est $O((n+k)\log n)$. Cette complexité est quasi optimale. En effet, la résolution des intersections d'un arrangement de segments implique le tri par comparaison, donc requiert au moins $O(n \log n)$. De plus, la simple énumération des intersections requiert $O(k)$. Aussi la complexité optimale est-elle $O(k+n\log n)$, ce qu'atteignent par ailleurs les algorithmes randomisés basés sur Bentley-Ottmann.

Concernant l'implémentation C++, on profite des concepts OO pour définir les différents objets géométriques. Étant donnée la nature rationnelle de toutes les coordonnées manipulées, on donne le choix à l'utilisateur de travailler avec deux types de nombre : des rationnels à base de **int** grâce à la bibliothèque `Boost.rational` ou bien des rationnels à précision arbitraire grâce à `Gnu MP` et le *wrapping* C++ qu'en fait `Gmpxx`).

Par ailleurs, la STL de C++ implémente std::set et std::map comme des arbres bicolores. J'ai donc profité de la modulabilité de ces structures (elles sont implémentés via des *templates*) afin d'implémenter l'arbre de recherche $\mathbf{T}$ et la file de priorité $\mathbf{Q}$ de l'algorithme 2.1.

On veut donc d'abord créer un std::set de segments (en fait de pointeur sur segments) avec une relation de comparaison modulable ($\prec_x$ change avec x) pour $\mathbf{T}$:

```
template <class T, class Comp> struct compare {
  Comp* comp; /*!< Evolving parameter*/
  compare(Comp* c) : comp(c) { };
  bool operator() (T* a, T* b) {
    return a->less(*b,*comp);
  }
};

template <class T, class Comp> struct BST {
  typedef std::set <T*, compare<T,Comp> > Type;
};
```

Étant donné la nature de l'attribut comp, agir sur la valeur pointée changera la relation de comparaison utile à l'arbre bicolore sous-jacent à std::set. L'intégrité de la structure ordonnée est maintenue par ce qui a été dit sur la Y-structure plus avant.

L'implémentation de $\mathbf{Q}$ se fait par une std::map d'events sans complication : on crée une classe template héritant de std::map à laquelle on ajoute simplement les quelques méthodes push, top et pop qui en donnent une utilisation plus classique en tant que file de priorité.

2.4 Complexité et performances

Étant donné le nombre important de segments présents dans $\mathcal{G}_{m,n}$ dès que m et n s'élèvent un peu (voir figure 5), la recherche de performance est tout aussi importante que le comptage même des (m,n)-cubes.

Évaluons la complexité $\gamma(m,n)$ du calcul de $\mathrm{Card}(\mathrm{M}^0_{m,n})$. Puisque l'algorithme de Bentley-Ottmann est en $O((N+K)\log N)$, avec N le nombre de segments et K le nombre d'intersections, il s'agit d'exprimer N et K en fonction de (m,n). Bien entendu, pour K on ne peut espérer trouver une expression exacte (sinon, tout ce que l'on fait ne sert à rien) et on se contentera d'un O.

Comme $K = O(N^2)$, il nous reste à exprimer N, c'est à dire le nombre de segment de l'ensemble S (cf. 2.2). Il est clair que les droites $D_{i,j,k}$ portant un segment de S (autre que les bords du carré) sont celles telles que $0 < k < i+j$. De plus, pour deux triplets (i,j,k) et (i',j',k'), $D_{i,j,k} = D_{i',j',k'}$ si et seulement si $\frac{1}{i\wedge j\wedge k}(i,j,k) = \frac{1}{i'\wedge j'\wedge k'}(i',j',k')$. Ainsi, S a un cardinal de

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=1}^{i+j-1} p(i,j,k)$$

où $p(i,j,k)$ vaut 1 si i,j et k sont premeirs entre eux, et 0 sinon. (Par exemple $p(i,j,k) = \lfloor \frac{1}{i\wedge j\wedge k} \rfloor$.)

Donc on a

$$\mathrm{Card}(S) \leq \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(i+j-1) = n\frac{m(m-1)}{2} + m\frac{n(n-1)}{2} - mn = mn\frac{m+n-4}{2}$$

Aussi a-t-on $N = O(mn(m+n))$, et comme $\log(mn(m+n)) = O(\log(m+n))$, on a finalement :

$$\gamma(m,n) = O\left(m^2n^2(m+n)^2\log(m+n)\right)$$

Les performances de mon implémentation avec utilisation du type **int** (auquel on peut se limiter la plupart du temps) sont comparables à celles de l'implémentation de `CGAL`. On peut donc raisonnablement dire que ces performances sont bonnes.

3 Génération des éléments de $\mathrm{M}^0_{m,n}$

Il peut être fort intéressant de générer rapidement tous les (m,n)-cubes étant donné m et n. En vision par ordinateur par exemple, il peut-être utile de disposer de tous les (m,n)-cubes pour pouvoir tester rapidement l'égalité de ceux-ci à certaines configurations. Nous allons ici montrer comment il est possible de le faire pour les (m,n)-cubes de $\mathrm{M}^0_{m,n}$.

Encore une fois, dans le cadre des graphes $\mathcal{G}^0_{m,n}$, le terme *face* désignera une face **bornée**.

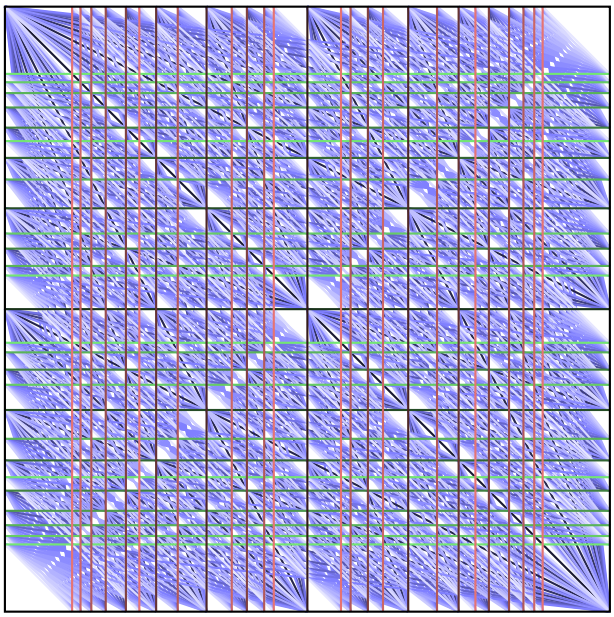


FIGURE 5 – $\mathcal{G}^0_{10,10}$: 695 segments

3.1 Idée générale

Soit $m, n \in \mathbb{N}^*$. On a montré que les faces de $\mathcal{G}_{m,n}$ étaient en bijection avec $\mathrm{M}^0_{m,n}$ (et c'est même ce qui a motivé la création de ce graphe). Aussi, afin de générer les (m,n)-cubes de $\mathrm{M}^0_{m,n}$, il est naturel d'exploiter cette bijection : parcourir les faces de $\mathcal{G}^0_{m,n}$ une et une seule fois et utiliser la bijection pour obtenir le (m,n)-cube correspondant (voir figure 6).

Se posent alors deux problèmes. Premièrement, le calcul de la bijection est lourd : étant donnée une face, il faut repérer ses arrêtes incidentes, calculer leur pente, repérer si la face est au-dessus ou en-dessous de la droite portant l'arrête, et enfin construire le (m,n)-cube associé à la face. En bref, cela fait beaucoup de calculs, ce qui demande de la ressource, surtout lors de la manipulation de rationnels à précision arbitraire. Deuxièmement, comment parcourir les faces ? Il faut y passer une et une seule fois, et aucun parcours n'est *a priori* préférable à un autre.

Il faut alors remarquer une propriété intéressante du graphe, qui relève de sa construction : deux faces adjacentes représentent deux (m,n)-cubes *proches*, au sens du nombre de voxels qui changent de l'un à l'autre. En effet, soit deux faces f et f' de $\mathcal{G}_{m,n}$ adjacentes selon une arrête e portée par la droite $D_{i,j,k}$: $ix+jy = k, i \wedge j \wedge k = 1$ associées respectivement à M et M' dans $\mathrm{M}^0_{m,n}$. On note $\mathbf{P} : z = -\lfloor \alpha x + \beta y \rfloor$ et $\mathbf{P'} : z = -\lfloor \alpha' x + \beta' y \rfloor$ des plans ayant respectivement M et M' pour (m,n)-motif en $(0,0)$. Disons, pour fixer les idées, que M est au-dessous de e et que M' est au-dessus. Alors :

$$\begin{aligned} \forall r \in \mathbb{N}^*, \lfloor \alpha' ri + \beta' rj \rfloor = rk = \lfloor \alpha ri + \beta rj \rfloor + 1 \\ \forall (x,y) \notin \{(ri, rj), r \in \mathbb{N}^*\}, \lfloor \alpha' x + \beta' y \rfloor = \lfloor \alpha x + \beta y \rfloor \end{aligned} \tag{1}$$

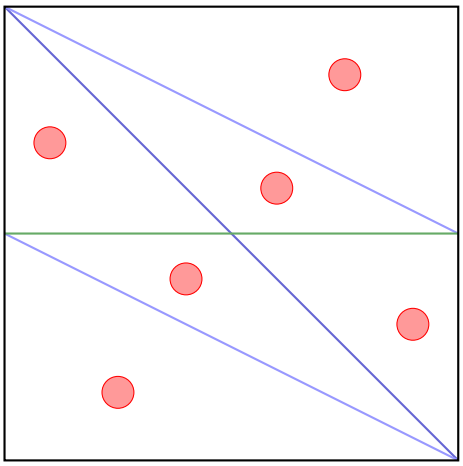


FIGURE 6 – $\mathcal{G}^0_{2,3}$, faces marquées

Nous avons donc maintenant un moyen de *passer* d'une face à une autre adjacente et de donner le (m,n)-cube de la face d'arrivée connaissant celui de départ **sans calculer explicitement la bijection**.

Le problème se réduit donc maintenant à :

- Connaître le (m,n)-cube associé à **une** face du graphe $\mathcal{G}^0_{m,n}$.
- Parcourir les faces du graphe $\mathcal{G}^0_{m,n}$ par adjacence.

C'est cette réduction que l'on exploite ci-après.

3.2 Graphe dual

Définition 3.1 (Graphe dual)**.** *Soit $G = (V, E)$ un graphe. On définit le graphe dual $G^* = (V^*, E^*)$ de G par :*

$$V^* = \{\textit{Faces de } G\}$$
$$E^* = \{(f, f') \in V^*, \exists e \in E, e \textit{ incidentes à } f \textit{ et } f'\}$$

${\mathcal{G}^0_{m,n}}^*$ (voir figure 7 pour la visualisation graphique) nous intéresse car un chemin de ce graphe est un chemin de parcours de faces par adjacence de $\mathcal{G}^0_{m,n}$. Un parcours sur tous les sommets de ${\mathcal{G}^0_{m,n}}^*$ répond donc partiellement (second point) à notre problème réduit du 3.1 en prenant un parcours de ce graphe.

De plus, parmi les faces de $\mathcal{G}^0_{m,n}$, il y en a une dont on connaît le (m,n)-cube associé sans aucun calcul : la face se situant sous toutes les droites $D_{i,j,k}, k > 0$ (visuellement, c'est celle *en bas à gauche*) a comme (m,n)-cube associé le (m,n)-cube *plat*, c'est-à-dire le (m,n)-motif en $(0,0)$ du plan $\mathbf{P}_0 : z = 0$. Cette remarque finit de répondre (premier point) au problème du 3.1.

En effet, il nous suffit maintenant de faire un parcours (par exemple en profondeur) du graphe dual ${\mathcal{G}^0_{m,n}}^*$ avec comme origine le sommet le plus bas (selon l'ordonnée) et de *flipper* les bons voxels (*i.e.* les monter ou les descendre d'une unité) lors du passage d'un sommet à un autre selon les égalités de (1).

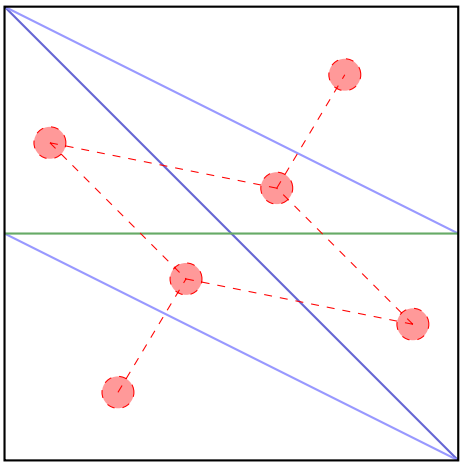


FIGURE 7 – $\mathcal{G}^0_{2,3}$ et ${\mathcal{G}^0_{2,3}}^*$

3.3 Codage en parité et algorithmes

Afin de ne pas gâcher les performances gagnées algorithmiquement avec des calculs importants pour le *flip* des voxels ou avec l'implémentation de la structure de (m,n)-motif en mémoire, il s'agit de trouver une représentation simple des (m,n)-motifs. Pour cela, on introduit le **codage en parité** des plans naïfs (on peut se référer à [Jam05] pour une présentation plus complète [1]).

Définition 3.2 (Codage en parité)**.** *Soit* $\mathbf{P} : z = -\lfloor \alpha x + \beta y + \mu \rfloor$ *un plan naïf. Le codage en parité de* $\mathbf{P}$ *est alors la suite bidimensionelle*

$$\mathcal{C}(\mathbf{P}) = (-\lfloor \alpha x + \beta y + \mu \rfloor \mod 2)_{(x,y)\in\mathbb{Z}^2}$$

On se convaincra que la donnée du codage en parité d'un plan naïf $\mathbf{P}$ et d'un point origine permet de reconstruire de manière unique $\mathbf{P}$.

Le codage $\mathcal{C}$ se restreint naturellement au (m,n)-motif en associant la sous-suite finie bidimensionnelle correspondante :

$$\mathcal{C}(\mathcal{R}_{m,n,\mathbf{P},a,b}) = (-\lfloor \alpha x + \beta y + \mu \rfloor \mod 2)_{\substack{a\leq x\leq a+m-1 \\ b\leq y\leq b+n-1}}$$

Ainsi, par la correspondance entre (m,n)-cubes et (m,n)-motif en $(0,0)$, on a une représentation simple des (m,n)-cubes comme matrices $m \times n$ de 0 et de 1.

Ceci fait, *flipper* devient une opération sans calcul gourmand. En effet, pour deux faces adjacentes f et f' de $\mathcal{G}^0_{m,n}$ dont l'arrête incidente e est portée par

1. Comme la définition le montre, le codage en parité est complètement indépendant du problème de la génération. Ce codage permet en fait d'aborder la problématique des plans naïfs de façon purement combinatoire. Le codage en parité d'un plan naïf est dit une suite bidimensionnelle de Rote et est l'équivalent des suites Sturmiennes dans le cadre des droites discrètes.

$D_{i,j,k} : ix + jy = k, i \wedge j \wedge k = 1$ et correspondants respectivement au (m,n)-cube $M \in \mathrm{M}^0_{m,n}$ de matrice $\mathcal{C}(M) = (m_{x,y})_{0 \leq x \leq m-1, 0 \leq y \leq n-1}$ et $M' \in \mathrm{M}^0_{m,n}$ de matrice $\mathcal{C}(M') = (m'_{x,y})_{0 \leq x \leq m-1, 0 \leq y \leq n-1}$, l'équation 1 montre que

$$\forall r \in \mathbb{N}^*,\ m'_{ri,rj} = \overline{m_{ri,rj}}$$
$$\forall (x,y) \notin \{(ri,rj), r \in \mathbb{N}^*\},\ m'_{x,y} = m_{x,y}$$

où $\overline{x} = 1 - x, \forall x \in \{0,1\}$.

On peut alors décrire l'algorithme de parcours du graphe dual ${\mathcal{G}^0_{m,n}}^*$. On suppose que chacun de ses sommets v stocke une matrice, initialement nulle, accessible *via* v.matrix, et un booléen accessible *via* v.discovered initialement `false` et que chacune de ses arrêtes e stocke les paramètres de pente i et j de la droite portant l'arrête correspondante dans $\mathcal{G}^0_{m,n}$, accesibles *via* $e.i$ et $e.j$. On effectue alors un parcours en profondeur du graphe ${\mathcal{G}^0_{m,n}}^*$ (par la méthode classique utilisant une pile ; cf. algorithme 3.2) en mettant à jour les matrices des sommets par la méthode de *flip* sus-décrite (cf. algorithme 3.1).

Algorithme 3.1 Flip

Paramètre(s): Une matrice source $(m_{x,y})_{0 \leq x \leq m-1, 0 \leq y \leq m-1}$, $i \in [\![0, m-1]\!]$, $j \in [\![0, n-1]\!]$, une matrice réceptrice $(m'_{x,y})_{0 \leq x \leq m-1, 0 \leq y \leq m-1}$.
Retour: $\emptyset$; la matrice réceptrice est le résultat du *flip* sur la matrice source.

```
for all (x, y) ∈ [[0, m − 1]] × [[0, n − 1]] do (* Copie *)
  m'_{x,y} ← m_{x,y}
end for
for (r = 1 ; ri < n and rj < m ; r ← r + 1) do (* Flip *)
  m'_{ri,rj} ← 1 − m_{ri,rj}
end for
```

3.4 Complexité et implémentation

Dans un graphe planaire connexe G où f est le nombre de faces (sans la face externe), e celui des arrêtes et s celui des sommets, on a par la formule d'Euler : $f = 1 + e - v$. De plus, on a $e \leq 3v - 6$. Donc finalement $f = O(v)$. Or, sur $\mathcal{G}_{mn}$, l'algorithme 3.2 à une complexité en temps de $\gamma(m,n) = O(fmn)$ (pour chaque face, on actualise les cases d'une matrice de taille $m \times n$). Comme $\mathcal{G}^0_{m,n}$ est un arrangement de N segments s'intersectant en K points, on a $v \leq N + K$. Ainsi, $\gamma(m,n) = O((N+K)mn)$. On a déjà vu au 2.4 que $N + K = O(m^2n^2(m+n)^2)$, donc

$$\gamma(m,n) = (m^3n^3(m+n)^2)$$

La complexité de l'algorithme de Bentley-Ottmann (cf. 2.4) préablablement appliqué afin d'obtenir le graphe est négligeable par rapport à $\gamma(m,n)$ (puisque $\log(m+n) = O(mn)$). Aussi la complexité finale est-elle en $O(m^3n^3(m+n)^2)$.

Ainsi, on génère les (m,n)-cubes de $\mathrm{M}^0_{m,n}$ dans un temps *a priori* légèrement plus important que le comptage simple.

L'implémentation est faite en C++ grâce à la bibliothèque `CGAL`. En effet, l'header `Arrangement_2.h` permet de gérer un graphe à partir d'un arrangement

Algorithme 3.2 DFS sur ${\mathcal{G}^0_{m,n}}^*$

Paramètre(s): Un sommet v_0 de ${\mathcal{G}^0_{m,n}}^*$.
Retour: Une liste **L** des matrices générées.

```
L ← ∅ (* Liste *)
S ← ∅ (* Pile *)

S ← S ∪ {v_0}, v_0.discovered ← true
while S ≠ ∅ do
  t ← pop(S)
  L ← L ∪ {t}
  for all e = (t, v) arrête de G^0_{m,n}* do
    if v.discovered = false then
      S ← S ∪ {v}
      v.discovered ← true
      Flip(t.matrix, e.i, e.j, v.matrix)
    end if
  end for
end while

return L
```

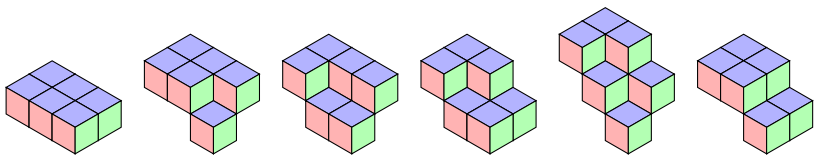


FIGURE 8 – Éléments de $\mathrm{M}^0_{2,3}$

de segments. On utilise également les fonctionnalités de `Arr_dcel_extend.h` qui permet d'associer une instance de classe aux sommets, arrêtes et faces du graphe sous-jacent à l'arrangement de segments. Bien que l'on aurait pu coder ces outils spécifiquement pour notre problème, `CGAL` donnent des garanties de corrections et performances qui en font donc ici une bibliothèque plus qu'adaptée.

Contrairement à la présentation théorique et algorithmique, on ne construit pas explicitement le graphe dual ${\mathcal{G}^0_{m,n}}^*$, étant donné que l'on peut travailler directement sur les faces de $\mathcal{G}^0_{m,n}$. Notre implémentation de l'algorithme 3.2 agit donc sur un dual implicite, profitant des outils de `CGAL` tels que les *handles* de faces (sorte de pointeurs sur les faces d'un graphe défini par un arrangement de segments) ou les *circulators* d'arrêtes (sorte d'itérateurs sur les arrêtes incidentes à une même face).

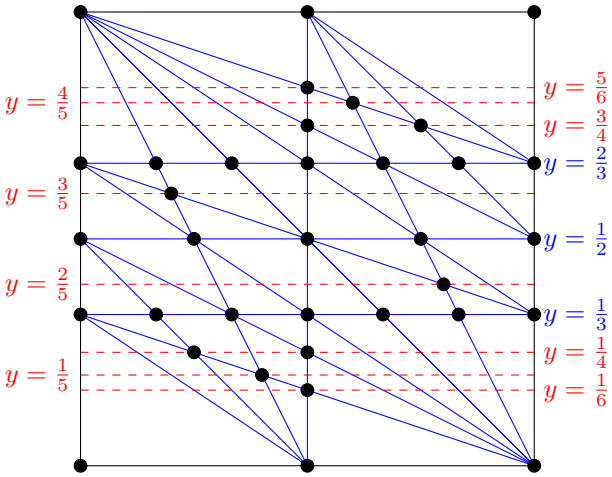
4 Avancées et ouvertures

4.1 Réduction sur $\mathcal{G}^0_{m,n}$

Le graphe $\mathcal{G}^0_{m,n}$ est manifestement des propriétés de symétries et de répétitions pourraient grandement réduire sa complexité. Le but de cette partie est de

montrer que l'étude d'une partie réduite du graphe permet la connaissance de $\mathcal{G}^0_{m,n}$ entier.

Illustrons cela sur $\mathcal{G}^0_{3,4}$.

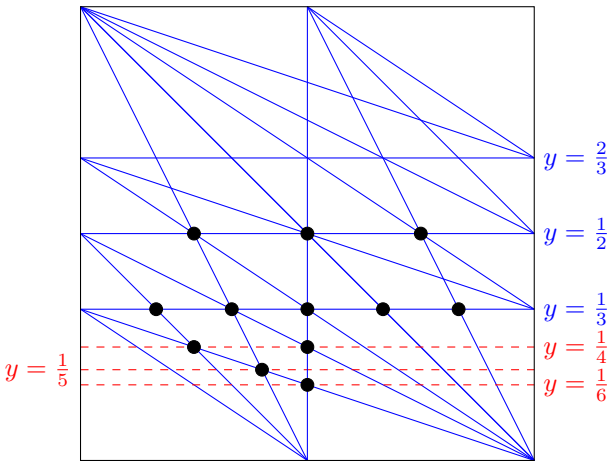


Il semblerait que chaque intersection d'une ligne horizontale $y = \frac{r}{d}, r \wedge d = 1$ ait son pendant sur toutes les lignes horizontales $y = \frac{r'}{d}, r' \wedge d = 1$.

Conjecture 4.1 (Association de sommets)**.** *Soient $m, n \in \mathbb{N}^*$ et $r, d \in \mathbb{N}^*$ tels que $0 < r < d \leq (m-1)(n-1)$ et $r \wedge d = 1$. Soient $\mathcal{S}_{1,d}$ l'ensemble des sommets de $\mathcal{G}^0_{m,n}$ sur d'ordonnée $\frac{1}{d}$ et $\mathcal{S}_{r,d}$ celui de ceux d'ordonnée $\frac{r}{d}$. Alors il existe une bijection de $\mathcal{S}_{r,d}$ dans $\mathcal{S}_{1,d}$ conservant le degré.*

Cette conjecture est facilement prouvable pour $d < n$ (c'est-à-dire pour les droites horizontales apparaissant effectivement dans l'arrangement de droites, bleues sur le dessin). Elle n'est actuellement pas prouvée pour $n \leq d \leq (m-1)(n-1)$ (droites rouges pointillées sur le dessin) mais semble relever plus de la technicité que du véritable challenge.

Ainsi, en admettant que la conjecture soit vraie, on peut grandement réduire le graphe : pour chaque $2 \leq d \leq (m-1)(n-1)$, il *suffit* de compter les sommets (avec multiplicité) de la droite d'équation $y = \frac{1}{d}$ puis de multiplier par $\varphi(d)$.



Cette simplification permet de réduire **significativement** le temps de calcul sur l'algorithme de comptage des faces de $\mathcal{G}^0_{m,n}$. Bien que la complexité soit la même (le nombre de droites horizontales du graphe est, avant comme après réduction, négligeable devant le nombre de droites obliques, qui lui, ne réduit asymptotiquement pas pusique l'on prend les droites $D_{i,j,k}$ pour $0 < k < i + \frac{j}{2}$ plutôt que pour $0 < k < i + j$ ce qui reste en $O(mn(m+n))$), le temps de calcul effectif s'en trouve bien amoindri, spécialement pour une expérience à m fixé (le nombre de droite retirée s'en ressent principalement par rapport à n). Par exemple, pour m=5, n=30, le résultat sort en $\sim$ 7s contre $\sim$ 20s sur le graphe entier.

Cette réduction du graphe peut également mener à la recherche d'une formule mathématique pour $\mathrm{Card}(\mathrm{M}^0_{m,n})$. Au courant du mois d'août, Éric Domenjoud a mené des travaux dans cette direction et a réussi à produire une telle formule. Nous ne la présenterons pas ici car elle est difficilement exploitable en l'état et demande un travail de simplification afin d'être abordable comme celle du théorème 1.1.

La démonstration de la formule repose sur la conjecture 4.1 et notamment sur un corrolaire immédiat de celle-ci : pour $m, n \in \mathbb{N}^*$, pour r, d tels que $0 < r < d \leq (m-1)(n-1)$, et pour $k \in \mathbb{N}$,

$$\mathrm{Card}\{s \in \mathcal{S}_{r,d}, \deg(s) = k\} = \mathrm{Card}\{s \in \mathcal{S}_{1,d}, \deg(s) = k\}$$

4.2 Cube dual

Tout le travail effectué a pris comme base $\mu = 0$ pour le décalage des plans naïfs considérés. Mais ce n'est pas tant la nullité de μ que sa contance qui nous importe. En effet, par le même raisonnement que dans la preuve du théorème 1.1, on peut définir le graphe $\mathcal{G}^\mu_{m,n}$ pour $\mu \in [0, 1[$. La figure 9 en donne un exemple. On a alors que l'ensemble $\mathrm{M}^\mu_{m,n}$ des (m, n)-cubes motif en $(0, 0)$ d'un plan de décalage μ est en bijection avec l'ensemble des faces de $\mathcal{G}^\mu_{m,n}$. On montre ici que l'exploitation de tous les $\mathcal{G}^\mu_{m,n}$ pour μ parcourant $[0, 1[$ permet le comptage de tous les (m, n)-cubes.

Avec les notations suivantes, pour $m, n \in \mathbb{N}^*$,
- $\mathrm{H}_{m,n} = [\![0, m-1]\!] \times [\![0, n-1]\!] \times \mathbb{Z}^3$.
- $\mathcal{P}$ l'ensemble des plans de $\mathbb{R}^3$ fonctionnels en z (*i.e.* de vecteur normal $\mathbf{v}$ tel que $\langle \mathbf{v}, \mathbf{e_3} \rangle \neq 0$).
- $\mathcal{Q} \subset \mathcal{P}$ l'ensemble des plans de $\mathbb{R}^3$ du type $\{(x, y, z) \in \mathbb{R}^3, \alpha x + \beta y + z + \mu\}$ avec $\alpha, \beta \in]0, 1[$ et $\mu \in [0, 1[$.

on définit l'opérateur dual $*$ sur $\mathbb{R}^3 \sqcup \mathcal{P}$ par :
- si $p = (a, b, c) \in \mathbb{R}^3$, alors $p^* \in \mathcal{P}$ est le plan d'équation $ax + by + z - c = 0$.
- si $P \in \mathcal{P}$ d'équation $\alpha x + \beta y + z + \mu = 0$, alors $P^* = (\alpha, \beta, -\mu) \in \mathbb{R}^3$.

Alors $\mathcal{Q}^* = \{P^*, P \in \mathcal{Q}\}$ est le cube $]0, 1[^2 \times [0, 1[$. Ainsi,

$$\mathbf{G} = [0, 1]^3 \cap \bigcup_{p \in \mathrm{H}_{m,n}} \left(p^* \cap [0, 1]^3\right)$$

est un polytope dont les volumes sont en bijection avec $\mathrm{M}_{m,n}$, l'ensemble de tous les (m, n)-cubes. (Nous ne détaillerons pas ce fait ici, cela suit exactement le raisonnement de [BP93].)

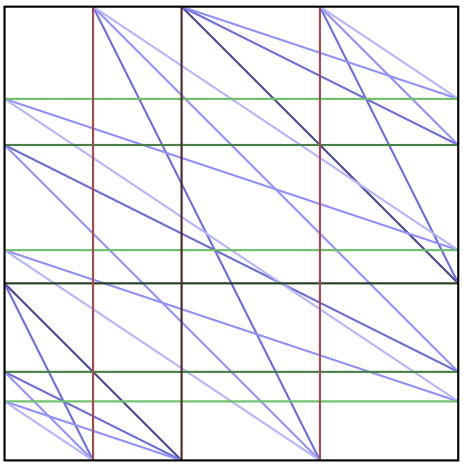


FIGURE 9 – $\mathcal{G}_{m,n}^{25/41}$

Il ne reste plus qu'à remarquer que pour $\mu \in [0,1]$, $\mathcal{G}_{m,n}^{\mu} = \mathbf{G} \cap P_\mu$ où P_μ est le plan d'équation $z = \mu$. Ainsi l'*empilement* des $\mathcal{G}_{m,n}^{\mu}$ pour μ parcourant $[0,1]$ en croissant *construit* $\mathbf{G}$. Les changements de nombre de faces de $\mathcal{G}_{m,n}^{\mu}$ quand μ varie donnent de l'information sur les volumes de $\mathbf{G}$. Ces changements donnent en fait toute l'information sur les volumes de $\mathbf{G}$ car $\mathbf{G}$ ne contient strictement aucun plan horizontaux (ceux que les $\mathbf{G}_{m,n}^{\mu}$ ne *détectent* pas) : en effet, un plan horizontal de $\mathbf{G}$ est un plan p^* pour $p = (0,0,c) \in \mathrm{H}_{m,n}$ et donc $c \in \mathbb{Z}$ impose $c = 0$ ou $c = 1$.

La figure 10 illustre l'empilement des $\mathcal{G}_{m,n}^{\mu}$ et l'information donnée sur les volumes de $\mathbf{G}$ pour $m = n = 3$.

L'expérimentation amène à deux conjectures. Si l'on note F_d la suite de Farey d'ordre $d \in \mathbb{N}^*$:

- Pour $\mu, \mu' \in [0,1] \backslash \mathrm{F}_{(m-1)(n-1)-1}$, alors $\mathcal{G}_{m,n}^{\mu}$ et $\mathcal{G}_{m,n}^{\mu'}$ ont le même nombre de faces. De plus, ce nombre de faces est plus grand que le nombre de faces de $\mathcal{G}_{m,n}^{\nu}$ pour $\nu \in \mathrm{F}_{(m-1)(n-1)-1}$.
- Pour r, d tel que $r \wedge d = 1$ et $\frac{r}{d} \in \mathrm{F}_{(m-1)(n-1)-1}$, $\mathcal{G}_{m,n}^{r/d}$ à le même nombre de faces que $\mathcal{G}_{m,n}^{1/d}$.

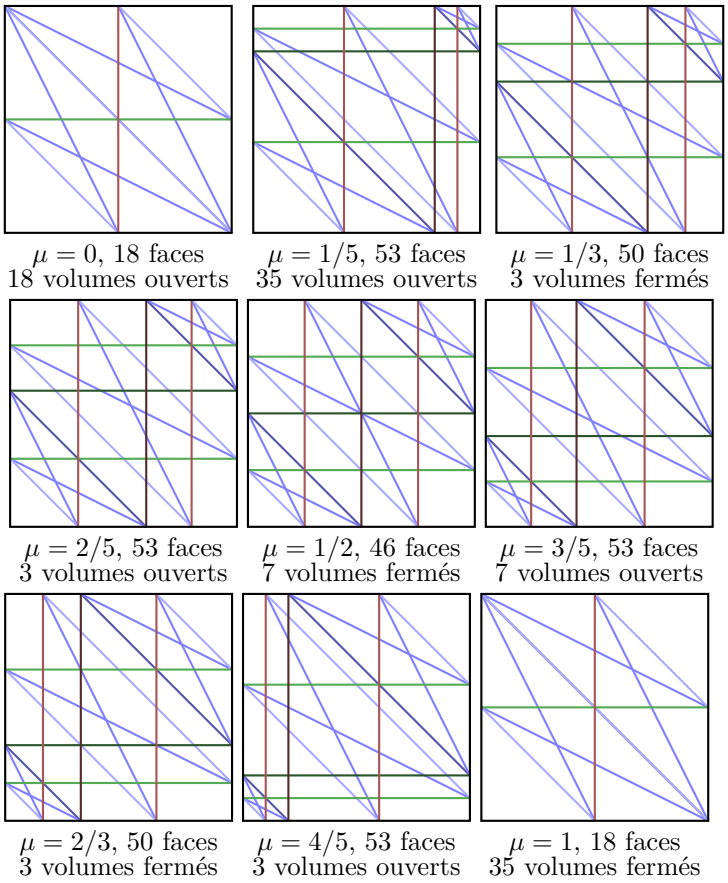
Ainsi, en notant $\mathcal{M}_{m,n}^{\mu} = \mathrm{Card}(\mathrm{M}_{m,n}^{\mu})$ (*i.e.* le nombre de faces de $\mathcal{G}_{m,n}^{\mu}$) et en profitant de la première conjecture, le nombre de volumes total est le nombre de volumes ouverts au cours du parcours du cube par les $\mathcal{G}_{m,n}^{\mu}$, donc :

$$\mathrm{Card}(\mathrm{M}_{m,n}) = \mathcal{M}_{m,n}^{f_1} + \sum_{i=1}^{k-1} (\mathcal{M}_{m,n}^{(f_{i+1}-f_i)/2} - \mathcal{M}_{m,n}^{f_i})$$

où $\mathrm{F}_{(m-1)(n-1)-1} = \{f_1 = 0, f_2, \ldots, f_k = 1\}$.

En profitant alors de la deuxième conjecture, on a :

$$\mathrm{Card}(\mathrm{M}_{m,n}) = \sum_{d=1}^{(m-1)(n-1)-1} \varphi(d) \mathcal{M}_{m,n}^{\infty} - \sum_{d=1}^{(m-1)(n-1)-1} \varphi(d) \mathcal{M}_{m,n}^{1/d} \qquad (2)$$

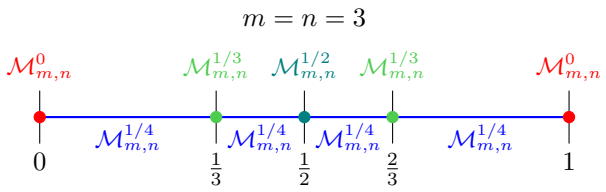


Reste à sommer les nombres de volumes ouverts (ou fermés, sans oublier les 18 fermés à la toute fin dans ce cas) pour connaître le nombre de volume total :

$$18 + 35 + 3 + 7 + 3 = 66$$

FIGURE 10 – Exploration de $\mathbf{G}$ par les $\mathcal{G}^{\mu}_{m,n}$.

où $\mathcal{M}^{\infty}_{m,n}$ est la valeur *générale* (bleu sur le dessin ci-dessous), c'est-à-dire par exemple $\mathcal{M}^{\infty}_{m,n} = \mathcal{M}^{1/((m-1)(n-1))}_{m,n}$.



Chaque couleur correspond à un nombre de faces du $\mathcal{G}^{\mu}_{m,n}$ correspondant.

La formule 2 permet une détermination expérimentale *rapide* du nombre total de (m,n)-cubes (la détermination des $\mathcal{M}^{\mu}_{m,n}$ se fait par l'algorithme de Bentley-Ottmann bien entendu). Cette détermination se fait en $O(mn\gamma(m,n))$ où $\gamma(m,n)$ est la complexité de l'algorithme de détermination du nombre de faces d'un seule graphe $\mathcal{G}^{\mu}_{m,n}$ (la complexité de la détermination des $\varphi(d)$ est négligeable). Finalement, la détermination de $\mathrm{Card}(\mathrm{M}_{m,n})$ se fait donc en $O(m^3n^3(m+n)^2\log(m+n))$ (voir 2.4).

La formule 2 est aussi un premier pas vers l'élaboration d'une formule effective du nombre de (m,n)-cubes. (Par exemple par une généralisation de la formule de Éric Domenjoud pour $\mathcal{M}^{0}_{m,n}$)

Références

[BL88] C.A. Berenstein and D. Lavine. On the number of digital straight line segments. *IEEE transactions on pattern analysis and machine intelligence*, pages 880–887, 1988.

[BO79] J.L. Bentley and T.A. Ottmann. Algorithms for reporting and counting geometric intersections. *Computers, IEEE Transactions on*, 100(9) :643–647, 1979.

[BP93] J. Berstel and M. Pocchiola. A geometric proof of the enumeration formula for sturmian words. *Internat. J. Algebra Comput*, 3(3) :349–355, 1993.

[DBCVK08] M. De Berg, O. Cheong, and M. Van Kreveld. *Computational geometry : algorithms and applications*. Springer-Verlag New York Inc, 2008.

[DJVV08] É. Domenjoud, D. Jamet, D. Vergnaud, and L. Vuillon. On the number of rectangular words of size $2 \times n$ of two-dimensional sequences with subword complexity $2m_1m_2$. In *Journées Montoises*, 2008.

[Jam05] Damien Jamet. *Géométrie discrète : une approche par la combinatoire des mots*. PhD thesis, Université Montpellier II, 2005.

[Mig91] F. Mignosi. On the number of factors of sturmian words. *Theoretical Computer Science*, 82(1) :71–84, 1991.

[RF91] J.P. Reveilles and J. Francon. Géométrie discrete, calcul en nombres entiers et algorithmique. 1991.

[SH76] M.I. Shamos and D. Hoey. Geometric intersection problems. In *17th Annual Symposium on Foundations of Computer Science*, pages 208–215. IEEE, 1976.