UNIVERSITÉ DU QUÉBEC À MONTRÉAL

UN ALGORITHME LINÉAIRE POUR LE CALCUL DE L'ENVELOPPE EXTERNE D'UN CHEMIN DISCRET

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN MATHÉMATIQUES

PAR

ROMAINE ARIANE WEBER

JANVIER 2015

[Cette page a été laissée intentionnellement blanche]

TABLE DES MATIÈRES

LISTE DES FIGURES

RÉSUMÉ

Ce mémoire concerne le problème du calcul de l'enveloppe externe d'une figure et se situe dans le domaine de la géométrie digitale, une branche importante de la géométrie discrète et algorithmique. Le point de vue adopté est intimement lié à la combinatoire des mots puisque toute figure discrète est codée par son contour, c'est-à-dire un mot sur l'alphabet de Freeman constitué de quatre lettres identifiant les quatre directions élémentaires sur le réseau carré $\mathbb{Z} \times \mathbb{Z}$, qui représente notre plan discret. Ainsi le problème se réduit à étudier les mots de contour. On présente donc un algorithme permettant de calculer l'enveloppe externe d'un chemin discret. On utilise une structure d'arbre quaternaire pour représenter les points du plan discret $\mathbb{Z} \times \mathbb{Z}$, enrichie par des liens de voisinages. En assimilant notre chemin discret à un labyrinthe, l'algorithme de calcul de l'enveloppe externe se base sur la méthode bien connue dite "de la main droite" permettant de sortir d'un labyrinthe efficacement. De plus, grâce à la structure d'arbre quaternaire, notre algorithme est linéaire en temps et en espace.

[Cette page a été laissée intentionnellement blanche]

INTRODUCTION

La géométrie discrète est la branche de la géométrie qui consiste à étudier non plus les figures comme des objets continus, mais comme des ensembles de pixels. Elle est tout indiquée pour travailler sur des écrans digitaux, et étant donné la présence toujours grandissante des supports numériques, il est devenu essentiel d'être le plus efficace possible dans ce domaine. La plupart des problématiques importantes de géométrie euclidienne le sont aussi en géométrie discrète, mais l'approche de celles-ci peut être tout à fait différente. Le cas qui nous intéresse dans le cadre ce mémoire est celui de la convexité.

En géométrie euclidienne, une figure est convexe si pour toute paire de point de la figure, le segment qui les relie est entièrement inclus dans celle-ci. Les objets convexes jouent un rôle important en géométrie, et dans d'autres domaines des mathématiques, comme l'analyse fonctionnelle, l'optimisation, les probabilités et la physique mathématique (Gruber, 2007). L'enveloppe convexe est la plus petite figure convexe contenant un ensemble de points. Un algorithme très connu pour calculer celle-ci a été découvert il y a assez longtemps, et est devenu un algorithme fondamental de la géométrie (Melkman, 1987). Depuis, des optimisations sur cet algorithme ont été découvertes (Yao, 1979; Chazelle, 1993) (Goodman et O'Rourke, 2004), et il a été réutilisé dans divers cadres (Sherali et Adams, 1990; Kim, 1992; Kim et al., 1997).

En géométrie discrète, sur des grilles carrées, utiliser la même définition de convexité que dans le cas euclidien n'est pas satisfaisant. En effet, avec cette définition, les seules figures convexes seraient les rectangles, ce qui est donc trop restrictif. Il importe de trouver une définition convenable de celle-ci. L'enveloppe convexe d'un sous-ensemble S de $\mathbb{Z}$ est l'intersection de tous les sous-ensembles euclidiens convexes contenant S. Une figure discrète F est dite convexe si son enveloppe convexe ne contient aucun point entier à l'extérieur de F. Pour calculer l'enveloppe convexe d'une figure discrète, on

peut simplement utiliser l'algorithme continu, mais ce n'est étonnamment pas toujours la méthode la plus efficace. En effet, Brlek, Lachaud et Provençal ont décrit un algorithme basé sur celui de Duval pour la factorisation en mots de Lyndon et sur la reconnaissance des mots de Christoffel pour reconnaitre les figures digitalement convexes (Duval, 1983; Brlek, Lachaud et Provençal, 2008). Ils ont également décrit un algorithme pour calculer l'enveloppe convexe basé sur la factorisation de Spitzer (Spitzer, 1956). Ces méthodes de résolution issues de la combinatoire des mots sont très éloignées de celles de la géométrie euclidienne car elles n'utilisent que des compteurs au lieu de nombres réels.

De par ces différences de méthodes, il est évident que les pré-requis des algorithmes ne sont pas les mêmes. En effet, si un ensemble de points suffit pour calculer une enveloppe convexe dans le plan euclidien, il est nécessaire d'avoir une figure simple et fermée pour la calcul de celle-ci dans le cas discret. Il s'agit en fait d'un pré-requis très courant pour les algorithmes de géométrie discrète (Brlek et Provençal, 2006).

Nous étudions dans ce mémoire la notion d'enveloppe externe d'une figure discrète, qui est le plus petit ensemble fermé simple contenant la figure. Pour cela, nous rappelons dans le chapitre 1 les bases nécessaires de combinatoire des mots en géométrie discrète, ainsi que quelques notions importantes pour la suite du mémoire. Ensuite, dans la section 2.1, nous définissons précisément ce qu'est l'enveloppe externe et rappelons les définitions de convexité et d'enveloppe convexe d'une figure discrète dans la section 2.3. Dans la section 3.2 nous présentons la structure d'arbre quaternaire issue de l'algorithme présenté dans (Brlek, Koskas et Provençal, 2011) pour la détection d'intersection dans un chemin discret en temps linéaire. Cette structure est nécessaire à l'algorithme principal du mémoire, après l'avoir un peu modifiée en y ajoutant des liens supplémentaires. En effet, nous allons utiliser les arbres quaternaires pour utiliser en quelque sorte "l'algorithme de la main droite", algorithme bien connu pour parcourir un labyrinthe. Enfin, après avoir expliqué et formalisé l'algorithme dans la section 3.3, nous allons l'utiliser dans la section 3.4 pour calculer l'enveloppe convexe d'une figure discrète.

L'algorithme exposé dans ce mémoire fut l'objet d'une communication à la 18^{e} conférence internationale "Discrete Geometry for Computer Imagery" (DGCI 2014), qui eut lieu à Sienne (Italie) du 10 au 12 septembre 2014, dans l'article "Efficient computation of the outer hull of a discrete path", rédigé en collaboration avec Hugo Tremblay, Jérôme Tremblay et de Srečko Brlek (Brlek et al., 2014).

[Cette page a été laissée intentionnellement blanche]

CHAPITRE I

PRÉLIMINAIRES

Au cours de ce chapitre, nous présentons les bases nécessaires de combinatoire des mots et de géométrie discrète nécessaires pour la suite. L'élément central de ce chapitre est la présentation du codage de Freeman. En effet, les travaux de Freeman ont permis de lier la combinatoire des mots et la géométrie discrète. Grâce à ce lien, les travaux de Spitzer en combinatoire appliqués aux probabilités ont pu être réutilisés dans la cadre de la géométrie discrète.

1.1 Généralités sur les mots

Soit Σ un ensemble fini de symboles appelés **lettres**. Pour cette raison, Σ est appelé **alphabet**. On appelle **concaténation** l'opération associative notée $\cdot$ et définie par :

$$\Sigma^n \times \Sigma^m \longrightarrow \Sigma^{n+m},$$
$$(w_1, w_2) \mapsto w_1 \cdot w_2.$$

où n et $m \in \mathbb{N}^*$.

Un **mot** w est une séquence finie, c'est à dire une suite ordonnée de lettres

$$w : [1...n] \longrightarrow \Sigma,$$

$$w = w_1 \cdot w_2 \cdot ... \cdot w_n,\ w_i \in \Sigma,\ 0 < i \leq n,$$

où w_i la ième lettre du mot, qu'on note parfois aussi $w[i]$. Par souci de lisibilité, on omet le $\cdot$, et on écrit donc $w = w_1 w_2 ... w_n$.

Un **sous-mot** u d'un mot w de longueur n est donné par un ensemble $I \subseteq [1, n]$ tel que $I = \{i_1, ..., i_k\}$, $u = w_{i_1} ... w_{i_k}$, on a donc $|u| = k$. L'ensemble I n'est pas nécessairement constitué d'indices consécutifs. Un **facteur** d'un mot est un sous-mot tel que ses indices sont consécutifs.

Le mot vide est noté ε. On dénote par $|w|$ la **longueur** du mot. Clairement $|\varepsilon| = 0$. On note $|w|_\alpha$ le nombre d'occurrences de la lettre $\alpha \in \Sigma$ dans le mot w.

On note Σ^n l'ensemble des mots de longueur n, $\Sigma^{\leq n}$ l'ensemble des mots d'au plus n lettres, et $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$ l'ensemble de tous les mots finis sur l'alphabet Σ, où $\Sigma^0 = \{\varepsilon\}$. On note $\Sigma^+ = \Sigma^* \setminus \Sigma^0$ l'ensemble des mots finis non vides.

Étant donné que l'opération de concaténation est associative, qu'elle possède un élément neutre ε, et qu'un mot $w \in \Sigma^*$ a une décomposition unique en éléments de Σ, Σ^*, muni de l'opération $\cdot$, est un **monoïde libre.**

Soient deux monoïdes M_1 et M_2, munis respectivement des opérations $\cdot$ et $*$, et ayant pour élément neutre m_1 et m_2. On appelle **morphisme** de $(M_1, \cdot, m_1)$ vers $(M_2, *, m_2)$ une application ϕ telle que :

(i) $\forall (x, y) \in M_1^2$, $\varphi(x \cdot y) = \varphi(x) * \varphi(y)$;

(ii) $\varphi(m_1) = m_2$.

Exemple 1.1.1. 1. *Soit un mot $w \in \Sigma^*$. Alors la longueur $|w|$ est un morphisme de monoïdes :*

$$| \, | : (\Sigma^*, \cdot, \varepsilon) \to (\mathbb{N}, +, 0),$$

car on a

$$|\varepsilon| = 0, \quad et \quad |uv| = |u| + |v|.$$

2. *Soit $\Sigma = \{a, b\}$. Alors l'échange de lettres $\overline{a} = b$ et $\overline{b} = a$ est un morphisme puisque $\overline{uv} = \overline{u} \cdot \overline{v}$. Un tel morphisme, qui va d'un monoïde vers lui-même, s'appelle un **endomorphismeL** L'échange de lettres est de plus une **involution**, c'est-à-dire qu'il*

est son propre inverse car $\overline{\overline{w}} = w$, $\forall w \in \Sigma^*$.

3. *Soit* ϕ *le morphisme tel qu'il associe chaque lettre de l'alphabet latin* A *à un entier naturel selon son positionnement dans l'ordre lexicographique. Alors on a :*

$$\phi : (A^*, \cdot, \varepsilon) \to (\mathbb{N}, +, 0),$$

tel que :

$$\phi(\varepsilon) = 0,\ \phi(l) = 12,\ \phi(a) = 1 \text{ et } \phi(la) = 13.$$

Pour tout mot $w \in \Sigma^*$, la k^e puissance de w est définie récursivement par $w^k = w^{k-1} \cdot w$, avec $w^0 = \varepsilon$. Un mot est dit **primitif** s'il n'est pas une puissance supérieure à 1 d'un mot non vide. Une **période** d'un mot est un nombre p tel que $w_i = w_{i+p}$ pour tout $1 \leq i \leq |w| - p$.

Exemple 1.1.2. *Prenons le mot primitif* $w = 0203$. *La deuxième puissance de* w *est* $w^2 = 02030203$, *et une période de* w^2 *est* $|w| = 4$.

Tout ordre total $<$ sur Σ, l'**ordre lexicographique** peut être étendu à tous les mots sur Σ^* de la manière suivante : soient $w, w' \in \Sigma^*$, alors,

$$w < w' \Leftrightarrow \text{(i)} w' = wv \text{ avec } v \in \Sigma^+,$$
$$\text{(ii)} w = uav \text{ et } w' = ubv' \text{ avec } a < b,\ a, b \in \Sigma,\ u, v, v' \in \Sigma^*.$$

On appelle $<$ l'ordre lexicographique.

Pour tout mot w, la fonction partielle $\Phi_w : \mathbb{N} \mapsto \mathbb{Z} \times \mathbb{Z}$ associe à tout entier $j \in [1, |w|]$ le vecteur $\Phi_w(j) = \overrightarrow{w_1 ... w_j}$.

Deux mots w, w' sont dit **conjugués**, et on écrit $w \equiv w'$, s'il existe $u, v \in \Sigma^*$ tels que $w = uv$ et $w' = vu$. La **classe de conjugaison** d'un mot est l'ensemble de tous ses conjugués, autrement dit l'ensemble de toutes les permutations circulaires de ses lettres.

Exemple 1.1.3. *Soient les mot* $w = 12031$ *et* $w' = 31120$. *On a* $w \equiv w'$, *et leur classe de conjugaison est* $\{12031, 20311, 03112, 31120, 11203\}$.

L'**image miroir** d'un mot $w = w_1w_2...w_n$ est le mot $\widetilde{w} = w_n...w_2w_1$. L'image miroir est aussi une involution, mais ça n'est pas un morphisme, mais un **antimorphisme** car $\widetilde{uv} = \widetilde{v} \cdot \widetilde{u}$. Un **palindrome** est un mot w tel que $w = \widetilde{w}$.

On peut généraliser l'exemple 1.1.1(2.) de la manière suivante : sur un alphabet donné Σ, on définit le **complément** $\overline{\alpha}$ d'une lettre α, telle que $\overline{\alpha} \in \Sigma$, $\overline{\overline{\alpha}} = \alpha$, et $\overline{\alpha_1 \cdot \alpha_2} = \overline{\alpha_1} \cdot \overline{\alpha_2}$. A noter que le complément est une permutation involutive sur un alphabet fini. Elle peut avoir ou ne pas avoir de points fixes.

Exemple 1.1.4. *Considérons l'alphabet* $\{0, 1, 2, 3\}$ *et l'involution sans point fixe définie par* $\overline{0} = 2$ *et* $\overline{1} = 3$. *Alors, si* $w = 01023$, *alors* $\overline{w} = 23201$.

La transformation $\widehat{w}$ est définie par composition entre l'image miroir et le complément :

$$\widehat{w} = \widetilde{\overline{w}} = \overline{\widetilde{w}}.$$

Cette transformation n'est pas un morphisme, mais un antimorphisme, étant donné que l'image miroir est un antimorphisme. Elle a une interprétation géométrique simple que l'on pourra voir dans la section 2.1.

1.2 Géométrie discrète et codage de Freeman

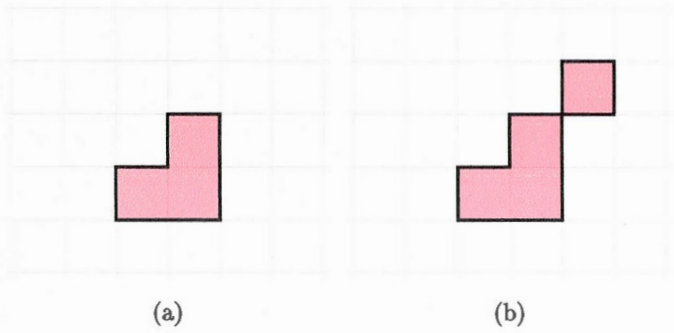
La géométrie discrète est une branche de la géométrie où l'on étudie des ensembles de **pixels**. Les pixels sont des carrés élémentaires de dimension 1 par 1 dans le plan euclidien, dont les sommets ont des coordonnées entières. Le plan discret est donc interprété comme l'ensemble des pixels.

Il y a une bijection entre les pixels du plan discret et $\mathbb{Z}^2$, en associant un couple $(a, b) \in \mathbb{Z}^2$ au coin inférieur gauche du pixel de coordonnées (a, b). Cette bijection permet de considérer un pixel comme un élément de $\mathbb{Z}^2$. Une **figure discrète** est un ensemble S de pixels dans le plan discret, c'est à dire $S \subset \mathbb{Z}^2$. Pour des raisons pratiques, nous ne nous intéresseront dans le cadre de ce mémoire qu'aux figures discrètes finies, c'est à dire celles dont la cardinalité $|S|$ est finie.

Soit un pixel $p = (p_x, p_y) \in \mathbb{Z}^2$. On définit l'ensemble $N_4(p)$ des **4-voisins** de p par $N_4(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$. L'ensemble $N_8(p)$ des **8-voisins** de p est défini par $N_8(p) = N_4(p) \cup \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}$.

Deux pixels p et q sont dits **α-adjacents** si $q \in N_\alpha(p)$, avec $\alpha \in \{4, 8\}$.

Une figure discrète est dite **α-connexe** si chaque pixel appartenant à la figure est α – *adjacent* à un autre pixel de la figure, avec $\alpha \in \{4, 8\}$.



(a) (b)

Figure 1.1: (a) Figure 4-connexe; (b) Figure 8-connexe

Dans (Freeman, 1961), Freeman propose une nouvelle manière d'encoder des figures discrètes sans trous, en utilisant le codage représenté dans la figure 1.2 : on considère l'alphabet $\mathcal{F} = \{0, 1, 2, 3\}$ où chacune des lettres représente un pas élémentaire dans le plan $\{\leftarrow, \uparrow, \rightarrow, \downarrow\}$. Donc, chaque lettre $\alpha \in \mathcal{F}$ correspond à une translation élémentaire des vecteurs respectifs $\vec{0} = (1, 0)$; $\vec{1} = (0, 1)$; $\vec{2} = (-1, 0)$ et $\vec{3} = (0, -1)$. Chaque mot w de $\mathcal{F}^*$ est associé à un vecteur $\vec{w}$ par le morphisme $\vec{-} : \mathcal{F}^* \longrightarrow \mathbb{Z} \times \mathbb{Z}$ où $\mathbb{Z} \times \mathbb{Z}$ est muni de l'addition vectorielle. Un **chemin discret** est une suite finie ou infinie de pas identifiés aux lettres de $\mathcal{F}$, c'est à dire un mot de $\mathcal{F}^*$.

L'alphabet $\mathcal{F}$ est appelé codage de Freeman. Il permet de définir entre autre certaines familles de figures discrètes. En effet, toute figure discrète sans trou peut être représentée par son contour, un chemin discret, et donc par un mot de $\mathcal{F}^*$.

On note $Step((x, y))$ le pas dans le codage de Freeman associé à une coordonnée :

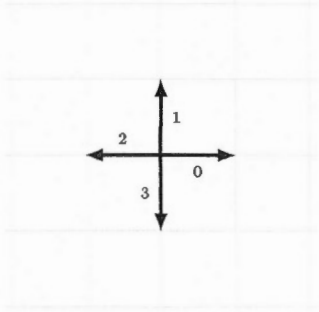
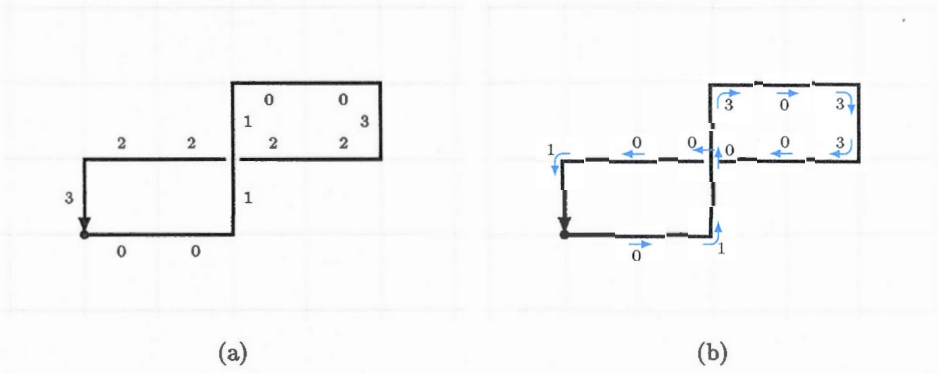


Figure 1.2: Les pas élémentaires du codage de Freeman

$Step((1,0) = 0$, $Step((0,1)) = 1$, $Step((-1,0)) = 2$ et $Step((0,-1)) = 3$.



(a) (b)

Figure 1.3: (a) Un chemin discret (b) sa suite de virages

Exemple 1.2.1. *Le mot* $w = 001100322223$ *représente le contour de la figure 1.3(a).*

Définition 1.2.2. *Un chemin discret* w *est* ***fermé*** *si son point de départ coïncide avec son point d'arrivée, c'est à dire que* $\vec{w} = \vec{0}$.

En terme de mots, la caractérisation suivante fournit un critère simple pour déterminer si un chemin codé par le mot w est fermé ou non.

Lemme 1.2.3. *Soit* $w \in \mathcal{F}^*$. *Le mot codé par* w *est fermé si et seulement si* $|w|_0 = |w|_2$ *et* $|w|_1 = |w|_3$.

Un chemin qui ne s'intersecte pas lui-même est dit **auto-évitant**. Si un chemin est

fermé et auto-évitant, la figure qu'il définit est simple. Dans le cas où le chemin se touche lui-même sans se couper il sera considéré comme auto-évitant, ainsi que dans le cas où le chemin revient sur lui-même (voir figure 1.4). La figure codée par la mot $w = 001100322223$ (figure 1.3) n'est pas simple, car le chemin qui la définit n'est pas auto-évitant.

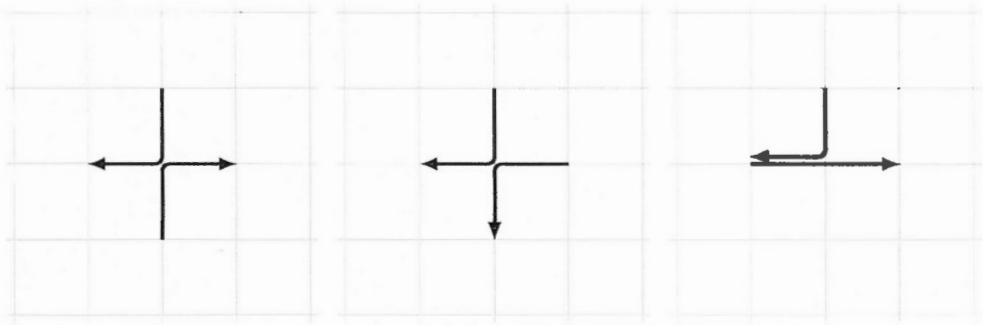


Figure 1.4: On considère que les chemins de (a), (b) et (c) ne se coupent pas

Ainsi, beaucoup d'algorithmes sur les figures discrètes, parfois plus efficaces que leurs analogues continus, on été découverts en utilisant les acquis de la combinatoire des mots, comme l'algorithme du calcul de l'enveloppe convexe que nous verrons dans la section 2.3, ou l'algorithme BKP qui permet, en temps linéaire, de déterminer si un chemin s'intersecte (Brlek, Koskas et Provençal, 2011). De plus certaines opérations géométriques sont plus efficaces à effectuer à partir du mot de contour de la figure. Par exemple, on peut aisément effectuer des opérations de rotation d'angle $k\frac{\pi}{2}$ par permutation circulaire des lettres dans l'alphabet. En effet, la rotation d'angle $\frac{\pi}{2}$ est réalisée par la permutation :

$$0 \to 1, 1 \to 2, 2 \to 3, 3 \to 0$$

dans $\mathcal{F}$. Il en va de même pour les autres rotations.

On peut également voir un chemin discret par la suite des virages effectués et non aux directions. Avec le codage de Freeman, il est facile de voir quels virages sont effectués en considérant les paires de lettres consécutives :

- 00, 11, 22, 33 correspondent à un pas en avant,
- 01, 12, 23, 30 correspondent à un virage à gauche,
- 03, 10, 21, 32 correspondent à un virage à droite,
- 02, 13, 20, 31 correspondent à un pas en arrière.

En calculant la différence modulo 4 entre les deux chiffres de chacune de ces possibilités, on a :

- $0-0=1-1=2-2=3-3=0 \mod 4$ pour un pas en avant,
- $1-0=2-1=3-2=0-3=1 \mod 4$ pour un virage à gauche,
- $3-0=0-1=1-2=2-3=3 \mod 4$ pour un virage à droite,
- $2-0=3-1=0-2=1-3=2 \mod 4$ pour un pas en arrière.

On peut donc créer un nouveau mot $\Delta(w)$ à partir du mot de contour w en parlant de virages et non plus de directions. Si on a un mot $w = w_1...w_n$, on l'obtient en posant :

$$\Delta(w) = (w_2 - w_1)(w_3 - w_2)...(w_n - w_{n-1}),$$

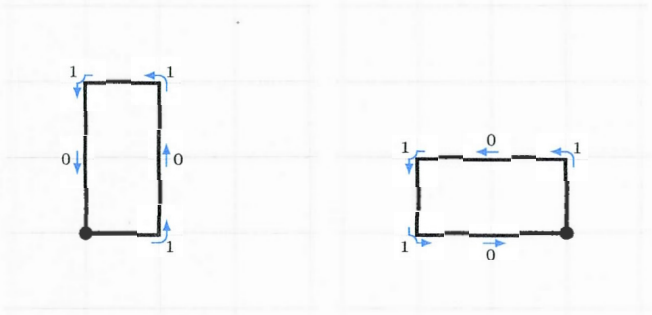
où les soustractions se font modulo 4. Le mot $\Delta(w)$ est appelé le **mot des premières différences de** w.

Exemple 1.2.4. *On voit dans la figure 1.3b que* $\Delta(w) = 01301101301$, *qui code les virages de* $w = 001100322223$, *et correspond à la même figure que la figure 1.3a.*

Il existe des cas où cette manière d'encoder les figures sera plus utile qu'avec les pas élémentaires du codage de Freeman. Par exemple, même après l'application d'une rotation sur une figure, son mot de première différence sera le même (voir figure 1.5).

Exemple 1.2.5. *Prenons la figure 1.5a codée par le mot* $w_1 = 011233$. *Après avoir effectué une rotation d'angle* $\frac{\pi}{2}$, *on obtient le mot* $w_2 = 122300$ *qui est représenté à la figure 1.5b. En calculant le mot des premières différences, on a* $\Delta(w_1) = \Delta(w_2) = 10110$.

Si un chemin discret est écrit dans le sens anti-horaire, les virages à gauches sont dit **saillants** et les virages à droite **rentrants** (de manière analogue, si le chemin est écrit dans le sens horaire, les virages à droite sont saillants et les virages à gauche sont rentrants).



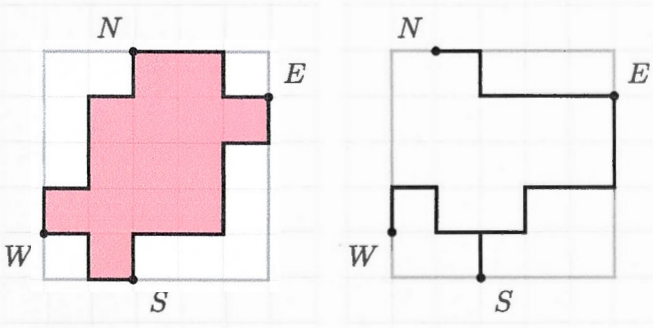
(a) (b)

Figure 1.5: (a) Une figure (b) la même figure après rotation

1.3 Boîte englobante et décomposition standard

Toute figure discrète w est contenue dans un **plus petit rectangle**, ou **boîte englobante**, en considérant les points extrémaux suivants :

- Le point sur le bord supérieur le plus à gauche est le point **N**
- Le point sur le bord gauche le plus en bas est le point **W**
- Le point sur le bord inférieur le plus à droite est le point **S**
- Le point sur le bord droit le plus en haut est le point **E**



(a) (b)

Figure 1.6: Boîte englobante de chemins

Ainsi, on peut définir **la décomposition standard du mot**. La décomposition standard d'un mot de contour est composée de quatre sous-mots, commençant et finissant

aux points extrémaux (voir la figure 1.6a). Si le chemin fermé w n'a pas de pas en arrière, la décomposition standard du mot est, en partant du point **W** dans le sens anti-horaire :

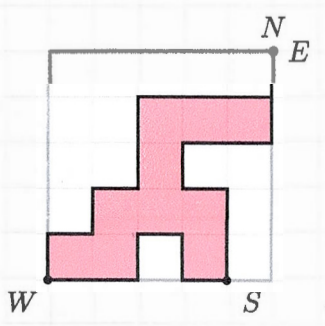
$$w = (0X) \cdot (1Y) \cdot (2Z) \cdot (3V),$$

où X, Y, Z et V sont soit vides (pas forcément tous en même temps), soit $X = x0$, $Y = y1$, $Z = z2$ et $V = v3$, avec $x, y, z, v \in \mathcal{F}^*$.

Exemple 1.3.1. *La figure 1.6a est codée par le mot* $w = 03010011012122323323$. *Sa décomposition standard est* $030 \cdot 1001101 \cdot 2122 \cdot 323323$ *dans le sens anti-horaire.*

Les points extrémaux peuvent être facilement calculés en temps linéaire en conservant les coordonnées extrêmes pendant la lecture du mot. L'algorithme 1 sert à trouver la position des points extrémaux dans un mot.

Dans certains cas extrêmes, des points extrémaux peuvent être confondus (voir la figure 1.7).



(a)

Figure 1.7: Figure où les points **N** et **E** sont confondus

On peut étendre de manière triviale la notion de boîte englobante aux chemins non fermés, afin de pouvoir trouver le point **W**. De plus si la figure n'est pas fermée, ou s'il y a des pas en arrière dans le mot, la première lettre n'est pas forcément 0, mais elle peut aussi être 1. En effet, la figure 1.6b n'est pas fermée, et elle est codée par le mot $w = 10303101001122212$.

Algorithm 1 Calcul de W

Entrée: Un mot $w \in \mathcal{F}^*$ codant un chemin discret

Sortie: La liste des positions des points W,S,E,N dans le mot w

1: $Wx = 0$ $Wy = 0$ $Wpos = 1$ //coordonnées et position du point W
2: $Sx = 0$ $Sy = 0$ $Spos = 1$ //coordonnées et position du point S
3: $Ex = 0$ $Ey = 0$ $Epos = 1$ //coordonnées et position du point E
4: $Nx = 0$ $Ny = 0$ $Npos = 1$ //coordonnées et position du point N
5: $curx = 0$ $cury = 0$ //coordonnées de la position courante
6: **pour** i de 1 à $|w|$ **faire**
7: **switch** $w[i]$ **faire**
8: **case** 0 : $curx++$; **break**
9: **case** 1 : $cury++$; **break**
10: **case** 2 : $curx--$; **break**
11: **case** 3 : $cury--$
12: **si** $curx < Wx$ **alors**
13: $Wx = curx$, $Wy = cury$, $Wpos = i$
14: **sinon si** $curx = Wx$ **and** $cury < Wy$ **alors**
15: $Wy = cury$, $Wpos = i$
16: **fin si**
17: **si** $cury < Sy$ **alors**
18: $Sx = curx$, $Sy = cury$, $Spos = i$
19: **sinon si** $cury = Sy$ **and** $curx > Sx$ **alors**
20: $Sx = curX$, $Spos = i$
21: **fin si**

22: **si** $curx > Ex$ **alors**
23: $Ex = curx$, $Ey = cury$, $Epos = i$
24: **sinon si** $curx = Ex$ **and** $cury > Ey$ **alors**
25: $Ey = cury$, $Epos = i$
26: **fin si**
27: **si** $cury > Ny$ **alors**
28: $Nx = curx$, $Ny = cury$, $Npos = i$
29: **sinon si** $cury = Ny$ **and** $curx < Nx$ **alors**
30: $Nx = curx$, $Npos = i$
31: **fin si**
32: **fin pour**
33: **renvoyer** $\{Wpos, Spos, Epos, Npos\}$

1.4 Mots de Lyndon et de Christoffel

Soit Σ un alphabet muni d'un ordre $<$. Il existe plusieurs caractérisations équivalentes des mots de Lyndon (Lothaire, 1997). Un mot de Lyndon est un mot tel qu'il est le plus petit dans sa classe de conjugaison, par rapport à la relation d'ordre $<$ sur Σ. En d'autres termes, un mot $l \in \Sigma^+$ est un mot de Lyndon s'il est primitif, et si

$$\text{pour tout } u, v \in \Sigma^* \text{ tels que } l = uv, \text{ on a } l < vu.$$

Théorème 1.4.1. *(Lothaire, 1997) Tout mot w admet une décomposition unique en mots de Lyndon décroissants :*

$$w = l_1^{n_1} l_2^{n_2} ... l_k^{n_k},$$

où $n_1, n_2, ..., n_k \geq 1$, et $l_1 > l_2 > ... > l_k$ sont des mots de Lyndon.

Plusieurs algorithmes ont été proposés pour le calcul de cette décomposition (Fredricksen et Maiorana, 1978; Duval, 1983; Lothaire, 2005). L'algorithme 2 calcule le premier facteur de Lyndon d'un mot. Le principe est de trouver le plus long préfixe u du mot w tel que u

est plus petit dans l'ordre lexicographique que tous ses suffixes. L'algorithme renvoie un couple : le premier élément est le premier mot de Lyndon, et le second est la puissance de ce mot, c'est à dire le nombre de fois où il est répété.

Algorithm 2 Premier facteur de Lyndon

Entrée: Un mot $w \in \Sigma^n$

Sortie: (l_1, n_1)

1: $(i, j) \leftarrow (1, 2)$
2: **tant que** $j \leq n$ et $w_i \leq w_j$ **faire**
3: **si** $w_i < w_j$ **alors**
4: $i \leftarrow 1$
5: **sinon**
6: $i \leftarrow i + 1$
7: **fin si**
8: $j \leftarrow j + 1$
9: **fin tant que**
10: **renvoyer** $(w_1 w_2 ... w_{j-i}, \lfloor (j-1)/(j-i) \rfloor)$

Supposons qu'à un moment de l'exécution, on a la factorisation $w = l^n pv$, où l est un mot de Lyndon, $n \geq 1$ le nombre de fois où l est répété, et p est un préfixe non vide de l. En d'autres termes, l'idée est de vérifier s'il y a une nouvelle puissance de l. Traduire cette étape de l'algorithme en valeurs de i et j donne $j - i = |l|$, $n = \lfloor (j-1)/(j-i) \rfloor$ et $j = |l^n p|$. Il y a trois cas possibles :

1. Si $w_{i+1} < w_{j+1}$, alors $l' = l^n p w_{i+1}$ est un nouveau mot de Lyndon. i repasse à 1 et on incrémente j. $(l', 1)$ devient la valeur courante de la paire (l_1, n_1). ;
2. Si $w_{i+1} = w_{j+1}$, alors p est remplacé par pw_{j+1} en tant que préfixe de l, puis on incrémente i et j ;
3. Si $w_{i+1} > w_{j+1}$, alors l'algorithme s'arrête et renvoie (l, n).

Cet algorithme est clairement linéaire en $n_1|l_1|$, donc la factorisation de Lyndon d'un mot est calculée en temps linéaire par rapport à $|w|$.

Exemple 1.4.2. *Considérons le mot* $w = 11010100101$. *Sa décomposition en mots de*

Lyndon est $11 \cdot 01^2 \cdot 00101$. *Les segments en rouge dans la figure 1.8 représentent cette décomposition.*

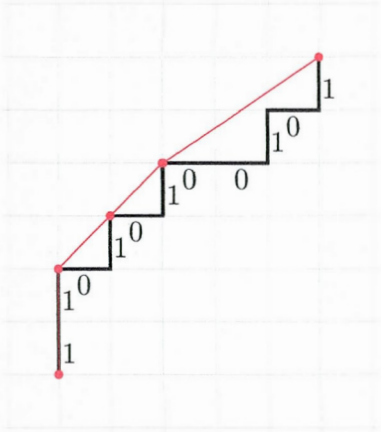


Figure 1.8: Un mot et sa décomposition de Lyndon

Soit $\Sigma = \{0, 1\}$. La **pente** d'un mot est définie par

$$\rho_\varepsilon = 1,\ \rho_w = |w|_1/|w|_0 \text{ pour } w \neq \varepsilon,$$

où on suppose que $1/0 = \infty$. Le nombre ρ correspond à la pente de la ligne droite qui relie le premier point au dernier point

Soient $k, n \in \mathbb{N}$ deux entiers premiers entre eux. Le **mot de Christoffel primitif** $C_{n,k} = w_1 ... w_n$ est défini par

$$w_i = \begin{cases} 0 & si \quad r_{i-1} < r_i, \\ 1 & si \quad r_{i-1} > r_i. \end{cases}$$

où r_i est le reste de la division de (ik) par n.

Exemple 1.4.3. *Soient* $k = 2$ *et* $n = 7$. *On obtient les valeurs suivantes :*

r_0	r_1	r_2	r_3	r_4	r_4	r_6	r_7
0	2	4	6	1	3	5	0

w_1	w_2	w_3	w_4	w_5	w_6	w_7
0	0	0	1	0	0	1

On dit qu'un mot est un mot de Christoffel s'il est une puissance d'un mot de Christoffel primitif.

Tout préfixe d'un mot de Christoffel a une pente la plus grande possible tout en restant inférieure ou égale à celle du mot entier. En conséquence, étant donné un mot de Christoffel u^r tel que $u^r = v^s$, u et v sont tous les deux aussi des mots de Christoffel.

Propriété 1.4.4. *Voici quelques propriétés des mots de Christoffel (Borel et Laubie, 1993; Brlek et al., 2009) :*

(i) *Tout mot primitif de Christoffel est aussi un mot de Lyndon.*

(ii) *Étant donnés deux mots de Christoffel c_1 et c_2 tels que $\rho_{c_1} = \rho_{c_2}$. Alors, on a soit $c_1 = c_2 \cdot w$, soit $c_2 = c_1 \cdot v$, avec $w, v \in \Sigma^*$.*

(iii) *Étant donnés deux mots de Christoffel primitifs c_1 et c_2, $c_1 < c_2$ dans l'ordre lexicographique si et seulement si $\rho_{c_1} < \rho_{c_2}$.*

(iv) *Étant donné $r \in \mathbb{Q}^+ \cup \{\infty\}$, soit F_r l'ensemble des mots $w \in \{0,1\}^*$ tels que $\rho(v) \leq r$ pour tout préfixe non vide v de w. F_r correspond aux mots définissant tous les chemins qui commencent à l'origine et qui sont en-dessous la ligne droite euclidienne de pente r. Parmi ces chemins, ceux qui sont les plus proches de la droite, et qui ont leur dernier point situé dessus sont des mots de Christoffel.*

Exemple 1.4.5. (i) *Le chemin 1.9 est codé par le mot de Christoffel $w =$* 00010010001001. *C'est la deuxième puissance du mot de Christoffel primitif $w' =$* 0001001. *On constate que w' est inférieur à toutes ses permutations par rapport à l'ordre lexicographique, il est donc le plus petit dans sa classe de conjugaison, c'est donc un mot de Lyndon.*

(ii) *$\rho_w = \frac{2}{5}$ et $\rho_{w'} = \frac{2}{5}$. On a donc $\rho_w = \rho_{w'}$, et $w = w' \cdot w'$.*

(iii) *Le mot de Christoffel $c =$ 01 a une pente $\rho_c = 1$. Dans l'ordre lexicographique, $c > w$, et $\rho_c > \rho_w$.*

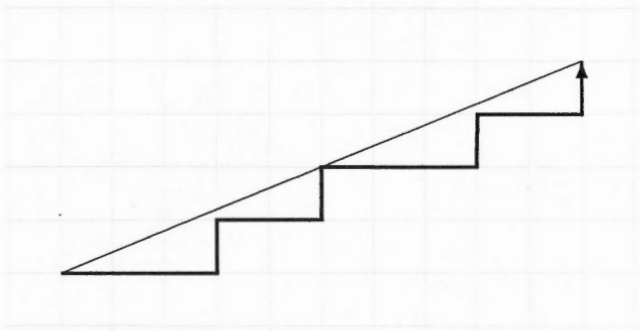


Figure 1.9: Un mot de Christoffel

(iv) *w' est un préfixe de w, on a $\rho_{w'} = \rho_w$ et le dernier point du chemin codé par w' est sur la droite de pente ρ_w, et w' est un mot de Christoffel.*

CHAPITRE II

ENVELOPPES EXTERNE ET CONVEXE

L'enveloppe externe est une notion nouvelle que nous allons introduire dans ce chapitre. Nous allons d'abord parler de l'enveloppe externe d'une figure discrète, puis nous allons pouvoir parler de l'enveloppe externe d'un chemin. Ensuite, nous allons rappeler la définition de l'enveloppe convexe dans le cas discret, et puis nous allons rappeler aussi l'algorithme de calcul de l'enveloppe convexe d'une figure discrète.

2.1 Enveloppe externe

L'enveloppe externe d'une figure discrète fermée S, notée Hull(S), est la plus petite figure discrète simple 8-connexe sans trous contenant S.

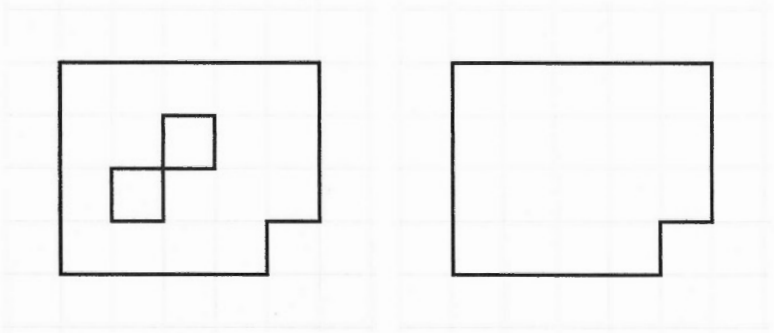


Figure 2.1: (a) Une figure discrète (b) Son enveloppe externe

Exemple 2.1.1. *Regardons la figure 2.1. La figure 2.1a est une figure discrète fermée*

quelconque. La figure 2.1b est le contour de la première, en retirant ce qu'il y a à l'intérieur. C'est son enveloppe externe.

En topologie, étant donné un ensemble S, le bord ∂S est l'ensemble des points de la fermeture de S.

On peut étendre la notion d'enveloppe externe à tout chemin discret P. On définit Euclidien$(P) \subset \mathbb{R}^2$ le plus petit sous-ensemble connecté simple de $\mathbb{R}^2$ contenant P. Hull(P) est alors le bord unique de Euclidien(P), c'est-à-dire Hull$(P) = \partial$(Euclidien(P)).

L'utilisation d'une figure euclidienne, et non plus discrète, dans cette définition permet de prendre en compte le cas d'un chemin non fermé, qui devient une figure euclidienne d'aire 0 (voir, par exemple 2.2), c'est à dire le chemin. Par définition, le bord d'un segment dans le plan euclidien $\mathbb{R}^2$ est le segment lui-même. Dans $\mathbb{Z}^2$, on exprime de tels cas en refermant simplement le chemin sur lui-même, en revenant sur nos pas. Si le chemin w est un chemin ouvert, auto-évitant et qui n'a pas de pas en arrière, son enveloppe externe sera $w \cdot \widehat{w}$. C'est à dire que, par exemple, le mot définissant l'enveloppe externe du segment défini par la lettre 0 dans le codage de Freeman sera 02.

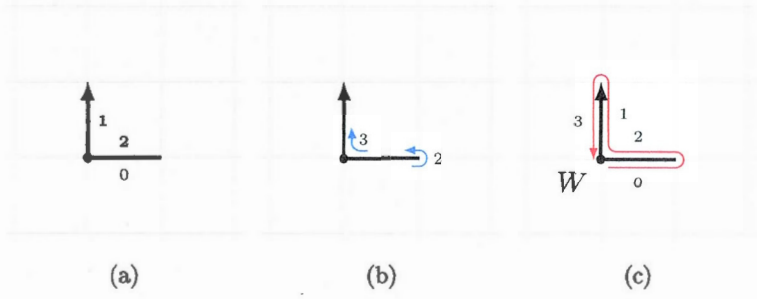


Figure 2.2: (a) Un chemin, (b) mot de première différence et (c) enveloppe externe

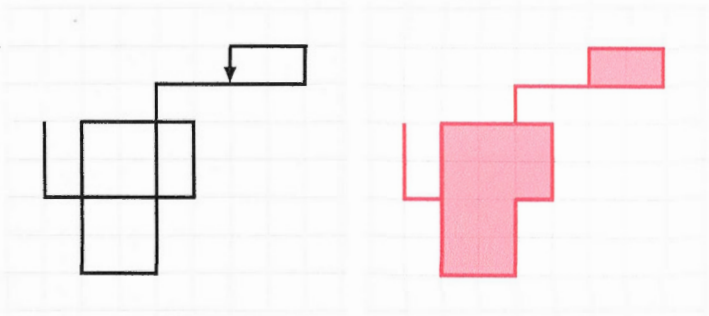
Considérons un chemin discret P. Alors Hull(P) est un chemin discret simple et fermé. Plus precisément, on a

Théorème 2.1.2. *(Brlek et al., 2014) Si P définit le contour d'une figure discrète F, alors* Hull(P) = Hull(F).

Démonstration. Par définition, $\partial(\text{Euclidien}(P))$ est un sous-ensemble de points de P. Par conséquent, c'est une union finie de segments de $\mathbb{R}^2$. Étant donné que toute union finie d'ensembles fermés est fermée, $\partial(\text{Euclidien}(P))$ est fermé. Montrons à présent que $\partial(\text{Euclidien}(P))$ est également simple. Soient $\overline{AB}$ et $\overline{CD}$ deux segments non orientés de $\partial(\text{Euclidien}(P))$ qui se coupent en un point x. Considérons les angles $(\overline{AxD})$ et $(\overline{CxB})$, qui n'ont que le point x en commun. Étant donné que les segments associés à ces angles forment deux chemins 8-connexes auto-évitant, $\partial(\text{Euclidien}(P))$ est simple. Enfin, si P définit le contour d'une figure discrète F, alors P est simple est fermé par définition. Par conséquent, $\text{Hull}(P) = P$, et comme $\text{Hull}(F)$ est le contour $\partial(F)$ par définition, on a

$$\text{Hull}(F) = \partial(F) = P = \text{Hull}(P).$$

■



(a) (b)

Figure 2.3: (a) Un chemin qui s'intersecte lui-même P ; (b) $\text{Euclidien}(P)$ et son bord (en rouge)

Il y a une bijection entre les chemins de $\mathbb{Z}^2$ et les mots codés dans l'alphabet de Freeman. Dans le cas des chemins fermés, tous les conjugués d'un mot de contour définissent une figure, seul le point de départ change. Grâce à cette relation, on peut assimiler un chemin P au mot qui le définit w, et on peut donc écrire $\text{Hull}(w)$ pour parler de $\text{Hull}(P)$ sans ambiguïté.

La principale utilité de l'enveloppe externe réside dans le fait qu'un grand nombre d'algorithme sur les figures discrètes nécessitent des chemins fermés auto-évitants. Le fait de pouvoir calculer l'enveloppe externe de chemins quelconques permet donc de généraliser ces algorithmes. Cela permet notamment de calculer l'enveloppe convexe de n'importe quel chemin discret.

2.2 Convexité digitale

Il existe plusieurs définitions de convexité digitale, dépendamment de si l'on veut que l'ensemble soit connecté ou non. Soit S une figure discrète 8-connexe. S est **digitalement convexe** si elle est la discrétisation de Gauss d'un sous-ensemble convexe R de $\mathbb{R}^2$. La discrétisation de Gauss consiste à retenir l'ensemble des points de $\mathbb{Z}$ appartenant ou étant à la frontière d'une figure continue. En d'autres termes, S est l'intersection du plan discret et d'une figure convexe R dans le plan euclidien : $S = \text{Conv}(R) \cap \mathbb{Z}^2$. Le pré-requis de le 8-connexité est nécessaire, car si on ne l'avait pas, les figures discrètes convexes ne seraient pas nécessairement connectées (voir figure 2.4). De plus, dans ce cas, on ne peut pas facilement vérifier si un sous-ensemble est convexe.

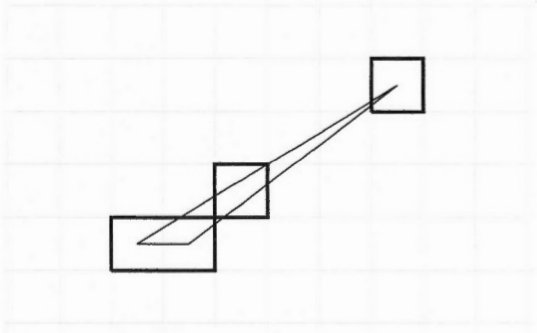


Figure 2.4: Une figure convexe et sa discrétisation

La convexité digitale peut aussi être vue d'un point de vue purement combinatoire : un mot w est digitalement convexe s'il est un mot codant le bord d'un sous-ensemble fini digitalement convexe de $\mathbb{Z}^2$. Un mot digitalement convexe est nécessairement fermé.

En géométrie euclidienne, **l'enveloppe convexe supérieure** de S, notée $\text{Conv}^+(S)$,

est la séquence des arêtes consécutives de Conv(S) orientées dans le sens horaire, en commençant par le sommet le plus bas, et finissant par le plus haut. De manière analogue, **l'enveloppe convexe inférieure** de S, notée $\text{Conv}^-(S)$, est la séquence des arêtes consécutives de Conv(S) orientées dans le sens horaire, en commençant par le sommet le plus haut, et finissant par le plus bas.

En utilisant la décomposition standard d'une figure discrète dans le sens horaire, telle que $w = w_1w_2w_3w_4$, le mot $w_1 \in \{0,1\}^*$ est **NW-convexe** si et seulement si il n'y a pas de point entier entre l'enveloppe convexe supérieure du vecteur issu du mot w et le chemin codé par w.

On définit la rotation circulaire dans le sens anti-horaire de $\frac{\pi}{2}$ par

$$\sigma : (0,1,2,3) \longrightarrow (1,2,3,0).$$

Alors on a que w_2 est **NE-convexe** si et seulement si $\sigma(w_2)$ est NW-convexe, et plus généralement, on a l'équivalence suivante

$$w_i \text{ est convexe} \iff \sigma^{i-1}(w_i) \text{ est NW-convexe.}$$

Il est évident que pour que w soit convexe, il faut que chaque w_i soit convexe, pour $i = 1,2,3,4$.

Théorème 2.2.1. *(Brlek, Lachaud et Provençal, 2008) Un mot w est NW-convexe si et seulement si son unique décomposition de Lyndon $w = l_1^{n_1}l_2^{n_2}...l_k^{n_k}$ est telle que tout ses l_i sont de mots de Christoffel primitifs.*

Lemme 2.2.2. *Soit $v \in \{0,1\}^*$ un mot codant un chemin NW-convexe, et soit e l'une des arêtes de son enveloppe convexe. Le facteur u de v tel que $\vec{u} = \vec{e}$ est un mot de Christoffel.*

Lemme 2.2.3. *Tout mot de Christoffel est NW-convexe.*

Démonstration. Soit v un mot codant un chemin NW-convexe, et soit la séquence d'arêtes $(e_1...e_k)$ le bord de son enveloppe convexe. Pour tout $i \in [1,k]$, soit u_i le facteur de

v déterminé par l'arête e_i tel que $v = u_i...u_k$. Soit l_i l'unique mot primitif tel que $u_i = l_i^{n_i}$.Par le lemme 2.2.2, u_i est un mot de Christoffel, ce qui implique que l_i est un mot primitif de Christoffel. Par la propriété i, l_i est également un mot de Lyndon. Par ailleurs, étant donné que $(e_i, ..., e_k)$ est l'enveloppe convexe de w, il en découle que la pente s_i de l'arête e_i est plus grande que la pente s_{i+i} de l'arête e_{i+1}, impliquant l'inégalité suivante :

$$\rho_{l_i} = \rho_{u_i} = s_i > s_{i+1} = \rho_{u_{i+1}} = \rho_{l_{i+1}}.$$

Étant donnée que l_i et l_{i+1} sont des mots de Christoffel, et que $\rho_{l_i} > \rho_{l_{i+1}}$, on en déduit par la propriété ii que $l_i > l_{i+1}$ dans l'ordre lexicographique. On a donc que $l_1^{n_1}...l_k^{n_k}$ est l'unique factorisation de w en mots de Lyndon décroissants.

Réciproquement, soit $v \in \{0,1\}^*$ tel que sa factorisation de Lyndon $l_1^{n_1}...l_k^{n_k}$ est composée de mots primitifs de Christoffel. Pour tout $i \in [1,k]$, soit e_i le segment reliant le point de départ du chemin codé par $l_i^{n_i}$ à son point d'arrivée. On veut montrer que $(e_1, ..., e_k)$ est l'enveloppe convexe supérieure de Φ_v. Étant donné que $l_i^{n_i}$ est un mot de Christoffel, le lemme 2.2.3 garantit que le chemin reste sous le segment. Par hypothèse on a $l_i > l_{i+1}$. En utilisant le même argument que précédemment, on a que la pente de e_i est strictement plus grande que celle de e_{i+1}. Nous venons de construire une séquence d'arêtes dont les sommets sont des points de Φ_v tout en restant au-dessus, et ayant des pentes décroissantes. $(e_1, ..., e_k)$ est donc exactement l'enveloppe convexe supérieure de Φ_v. Enfin, pour tout $i \in [1,k]$, étant donné que $l_i^{n_i}$ est un ot de Christoffel par le lemme 2.2.3, il n'y a pas de point entier entre e_i et $\Phi_{l_i^{n_i}}$. Il en résulte qu'il n'y a pas de point entier entre Φ_v et son enveloppe convexe supérieure, ce qui implique que v est NW-convexe. ■

Pour vérifier si un mot est NW-convexe, il faut donc d'abord calculer sa factorisation en mots de Lyndon (voir algorithme 2 dans la section 1.4 du chapitre précédent), puis vérifier si ce sont des mots de Christoffel, en se basant sur la première définition de ceux-ci (Christoffel, 1875).

Considérons l'ensemble Alph(w) des lettres de w, ainsi que sa décomposition standard $w_1w_2w_3w_4$. Si pour un certain $i \in \{1,2,3,4\}$, w_i contient plus de 2 lettres, ce qui veut dire $\text{Alph}(\sigma^{i-1}(w_i)) \not\subseteq \{0,1\}$, alors w n'est pas convexe.

2.3 Enveloppe convexe d'une figure discrète

L'enveloppe convexe de S, notée Conv(S) est la figure euclidienne convexe de plus petite surface contenant S. Dans le cas d'un chemin discret fermé auto évitant w, il est possible de calculer son enveloppe convexe Conv(w) avec sa factorisation de Spitzer (Brlek et al., 2009; Spitzer, 1956), qui est définie plus bas.

Soit un morphisme $\Psi : \{0,1\}^* \to \mathbb{R}$, où $\mathbb{R}$ est vu comme un monoïde additif. Alors pour tout $r \in \mathbb{R}$, on définit les ensembles

$$C_r = \{v \in \{0,1\}^+ \mid \Psi(v) = r|v|\}$$

ainsi que

$$B_r = C_r \Big\backslash \Big(\bigcup_{s \geq r} C_s \cdot \{0,1\}^+ \Big)$$

Exemple 2.3.1. *Pour Ψ défini par $\Psi(0) = -1$ et $\Psi(1) = 1$, le mot $v = 00100011$ est dans $B_{-1/4}$. On a $\Psi(00100011) = -2$, $|v| = 8$. On a $r = \Psi(v)/|v| = -2/8 = -1/4$, et donc $v \in C_{-1/4}$. De plus, tous ses préfixes sont dans C_s tels que $s \leq -1/4$: 0 et 00 sont dans C_{-1}, 001 est dans $C_{-1/3}$,001000 est dans $C_{-2/3}$, 0010 est dans $C_{-1/2}$, 0010001 est dans $C_{-3/7}$ et 00100 dans $C_{-3/5}$.*

En revanche, le mot 00011001 est dans $C_{-1/4}$, mais pas dans $B_{-1/4}$, car son préfixe 00011 est dans $C_{-1/5}$.

Géométriquement, l'ensemble C_r représente un ensemble de mots dont la pente est la même : pour $v \in C_0$, on a $\rho_v = 1$, pour $w \in C_1$, on a $\rho_w = \infty$, et pour $x \in C_{-1}$, on a $\rho_x = 0$. L'ensemble B_r représente les mots de C_r tels qu'ils restent toujours sous la pente associée aux mots de l'ensemble C_r (voir figure 2.5).

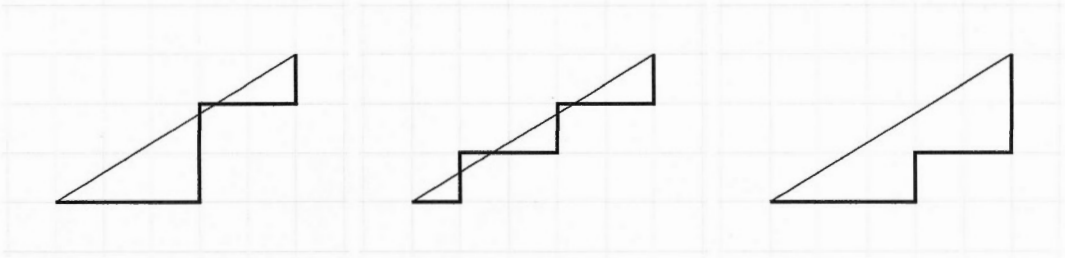


Figure 2.5: Ces trois chemins sont dans $C_{-1/4}$, mais seul le troisième est dans $B_{-1/4}$

Théorème 2.3.2. *(Spitzer, 1956) Tout mot w sur $\{0,1\}^*$ admet une décomposition unique en $w = b_1 b_2 ... b_k$ où $b_i \in B_{r_i}$ pour $i = 1, 2, ..., k$, et si $i \neq k$, alors $r_i \geq r_{i+1}$.*

En prenant le morphisme Ψ défini par $\Psi(0) = -1$ et $\Psi(1) = 1$, la séparation $b_i \cdot b_{i+1}$ entre chaque paire de facteurs consécutifs telle que $r_1 > r_{i+1}$ correspond exactement aux sommets de l'enveloppe convexe du chemin codé par w. (Brlek, Lachaud et Provençal, 2008).

Notons que la pente d'un mot $v \in B_r$ est $\rho_v = (1+r)/(1-r)$, et donc si $v \in B_r$ et $v' \in B_{r'}$ alors

$$r < r' \Longleftrightarrow \rho_v < \rho_{v'}.$$

De plus, la famille des ensembles $(B_r)_{r \leq 0}$ avec $\Psi(0) = -1$ et $\Psi(1) = +1$ a la propriété suivante : soient $u \in B_r$ et $v \in B_t$. Si $r < t$, alors le mot uv appartient à B_s pour $r < s < t$.

Étant donnée une figure $w = w_1 w_2 ... w_n \in \{0,1\}^*$, on peut calculer la partie NW de la décomposition de Spitzer grâce à l'algorithme 3.

On peut voir que si le nombre d'occurrences de 0 et de 1 est stocké, la factorisation se fera en temps linéaire. De plus, si on utilise une pile pour le calcul, l'algorithme obtenu devient une version discrète de l'algorithme classique de Melkman (Melkman, 1987) pour la construction de l'enveloppe convexe d'un polygone en temps linéaire, bien qu'on ne travaille que sur la partie NW de l'enveloppe au lieu d'appliquer l'algorithme sur le

Algorithm 3 Décomposition de Spitzer

Entrée: Un mot $w \in \Sigma^n$

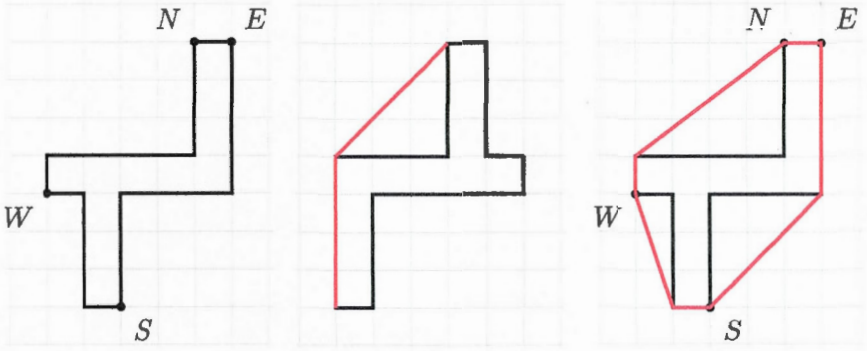
Sortie: une liste de mots b

```
1: (b_1, b_2, ..., b_n) ← (w_1, w_2, ..., w_n)
2: continue ← vrai
3: tant que continue = vrai faire
4:     continue ← faux
5:     pour i de 1 à |b| faire
6:         si ρ(b_i) < ρ(b_{i+1}) alors
7:             b_i ← b_{i+1}
8:             continue ← vrai
9:             i ← i + 1
10:        fin si
11:    fin pour
12: fin tant que
13: renvoyer b
```

polygone entier. De plus, cette méthode ne prend pas en compte les chemins qui ne sont pas sur $\{0,1\}^*$. Les parties NE, SE et SW se calculent de la même manière, après avoir effectué des rotations successives sur le mot.

Exemple 2.3.3. *La figure 2.6 est codée par le mot* $w = 10000111033332223332111 2 = w_1 \cdot w_2 \cdot w_3 \cdot w_4$ *tel que* $w_1 = 10000111$, $w_2 = 0$, $w_3 = 3333222333$ *et* $w_4 = 21112$. *On a alors* $w_1 = 1 \cdot 0000111$, $\sigma^1(w_2) = 1$, $\sigma^2(w_3) = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 000111$ *et* $\sigma^3(w_4) = 1 \cdot 0001$. *Dans la figure 2.6, les lignes rouges correspondent aux vecteurs issus des sous-mots de la factorisation.*

L'algorithme 4 issu de l'article de (Brlek et al., 2009) permet de calculer l'enveloppe NW-convexe d'une figure discrète dans $\{0,1,2,3\}^*$. Pour cet algorithme, on crée une nouvelle structure SousMot, qui est un triplet (i,a,b), où i représente la position de départ du sous-mot dans le mot d'origine, a est le déplacement vertical par rapport au



(a) (b) (c)

Figure 2.6: (a) Un chemin w (b) $\sigma^2(w)$ ainsi que la factorisation de Spitzer de $\sigma^2(w_3)$ (c) la factorisation de Spitzer de w

point de départ du sous-mot dans la figure et b est le déplacement horizontal par rapport au point de départ du sous-mot dans la figure.

De la même manière que précédemment, pour avoir la totalité de l'enveloppe convexe d'une figure, il faut calculer l'enveloppe NW-convexe de la figure, puis effectuer une rotation pour calculer la partie suivante.

Algorithm 4 Calcul de l'enveloppe NW-convexe

Entrée: Un mot $w \in \Sigma^n$

Sortie: P une pile de SousMots

1: **Var** u, v deux SousMots
2: **pour** i de 1 à n **faire**
3: $u.i \leftarrow i$
4: **switch** $w[i]$ **faire**
5: **case** 0 : $u.a \leftarrow 0$; $u.b \leftarrow 1$; **break**
6: **case** 1 : $u.a \leftarrow 1$; $u.b \leftarrow 0$; **break**
7: **case** 2 : $u.a \leftarrow 0$; $u.b \leftarrow -1$; **break**
8: **case** 3 : $u.a \leftarrow -1$; $u.b \leftarrow 0$; **break**
9: $l \leftarrow$ vrai
10: **tant que** l and (P n'est pas vide) **faire**
11: $v \leftarrow sommet(P)$
12: **si** $(v.a * u.b \leq u.a * v.b)$/*Si le vecteur u est à la gauche du vecteur v, on concatène les deux*/ **alors**
13: $u.i \leftarrow v.i$
14: $u.a \leftarrow u.a + v.a$
15: $u.b \leftarrow u.b + v.b$
16: $dpiler(P)$
17: **sinon** $l \leftarrow$ faux
18: **fin si**
19: **fin tant que**
20: $empiler(P, u)$
21: **fin pour**
22: **renvoyer** P /*P est donc une pile qui contient les vecteurs composant l'enveloppe NW-convexe du mot.*/

[Cette page a été laissée intentionnellement blanche]

CHAPITRE III

L'ALGORITHME

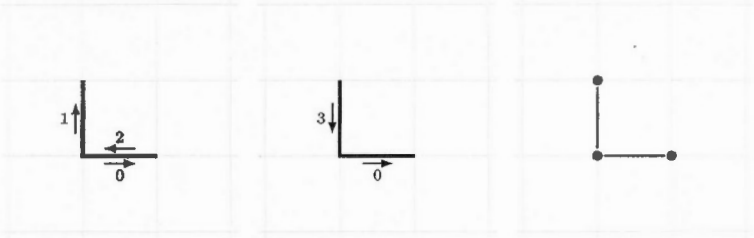
L'algorithme de calcul de l'enveloppe d'un chemin discret est l'élément central de ce mémoire. L'idée de base de l'algorithme est la même que la méthode utilisée pour sortir d'un labyrinthe. En effet, en tournant systématiquement dans la même direction, on en trouve la sortie. L'idée du calcul de l'enveloppe externe est la même : en se plaçant sur un point appartenant déjà à l'enveloppe externe, et en tournant toujours dans la même direction, on se retrouve au point de départ en ayant parcouru l'enveloppe externe du chemin. La structure utilisée est celle d'arbre quaternaire, issue de l'article (Brlek, Koskas et Provençal, 2011), où est présenté un algorithme pour détecter si un chemin discret s'intersecte. Cet algorithme est la base utilisée pour calculer l'enveloppe externe d'un chemin.

3.1 Principe

Soit $w \in \mathcal{F}^*$ un chemin discret et soit $G_w = (V, E)$ son graphe sous-jacent, où V est l'ensemble des sommets de w et E est l'ensemble des arêtes de w. On constate que l'application $g : w \mapsto G_w$ n'est pas bijective, étant donné qu'elle n'est pas injective (voir figure 3.1).

G_w admet la topologie suivante $\mathcal{T}(G_w)$: identifier toute arête au rectangle $[0, 1] \times [0, \delta]$ où $\delta \in \mathbb{R}$ est petit et les coller aux sommets coïncidents, en considérant que l'intersection de ces rectangles est dans $\mathcal{T}(G_w)$. De manière intuitive, cette topologie est obtenue en grossissant les arêtes de G_w, comme dans la figure 3.2. On peut montrer que $\mathcal{T}(G_w)$ est

fermée sous des unions arbitraires et des intersections finies telles qu'elles en font une topologie. De plus, la connectivité de G_w implique celle de $\mathcal{T}(G_w)$ (Brlek et al., 2014). Pour avoir plus d'informations sur la topologie en théorie des graphes, voir (Vella, 2005).



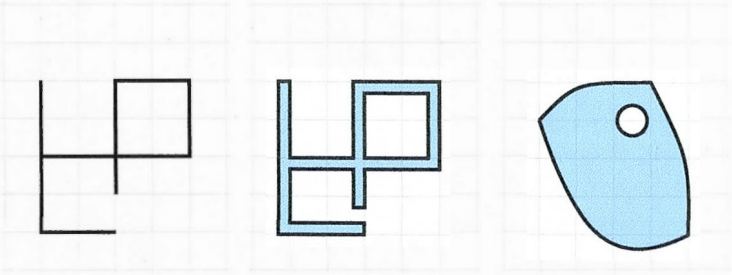
(a) (b) (c)

Figure 3.1: (a) et (b) deux chemin ayant la même représentation graphique (c) leur graphe sous-jacent

Avec cette topologie, G_w peut être alors vu comme un labyrinthe. $\mathcal{T}(G_w)$ est homéomorphe à une région connexe R, comme dans la figure 3.2c. Le problème de la détermination de l'enveloppe externe de G_w est ainsi équivalent à décrire le bord $\delta(R)$ de R. Maintenant, étant donné un départ sur $\delta(R)$, la solution peut être trouvée en utilisant la technique dite "de la main droite" pour traverser les labyrinthes. En appliquant cette règle, l'idée de l'algorithme est de marcher le long du chemin, en commençant par un point d'origine sur l'enveloppe externe, et en tournant systématiquement à droite à chaque intersection et en retournant au point d'origine. La discussion précédente garantit que le résultat sera exactement l'enveloppe externe de w.

Pour implémenter efficacement cette procédure, quelques problèmes doivent être résolus. En premier lieu, le point de départ doit être à coup sûr situé sur l'enveloppe externe. En effet, si ça n'était pas le cas, le résultat pourrait ne pas décrire la bonne figure. Ce problème est résolu assez facilement en choisissant la coordonnée d'un des points extrémaux associés au mot de contour w comme point de départ. Nous prenons systématiquement W comme point de départ.

Ensuite, dans le cas où le chemin revient sur le point W avant de continuer (Le chemin



(a) (b) (c)

Figure 3.2: (a) Un chemin (b) le labyrinthe associé à ce chemin par la topologie $\mathcal{T}(G_w)$ et (c) une région connexe homéomorphe R

codé par le mot 021 en est l'exemple le plus simple, voir la figure 2.2) , on doit être certain que l'algorithme ne s'arrête pas tant que tous les sous-chemins n'ont pas été explorés. La solution la plus simple est de sauvegarder une liste de tous les voisins de W. De par le fait que c'est un point extrême, il ne peut pas y avoir de voisin à la gauche ou en-dessous de W, donc cette liste ne peut avoir qu'au plus deux éléments.

Enfin, il est nécessaire de trouver les intersections et de décider le virage le plus à droite. Afin de garder l'information sur les relations de voisinage dans la figure, nous allons utiliser la structure **d'arbre quaternaire**.

3.2 Arbres quaternaires

Introduite dans (Brlek, Koskas et Provençal, 2011), pour détecter les intersections dans un chemin discret, la structure d'arbre quaternaire permet de représenter les points du plan discret à l'aide de leurs cordonnées.

Supposons tout d'abord que le chemin est codé par un mot w commençant à l'origine $(0,0)$ et inclus dans le premier quadrant $\mathbb{N} \times \mathbb{N}$, ce qui signifie que toutes les coordonnées de la figure sont positives. Il est possible de modifier la structure pour ne plus avoir cette contrainte. Dans $\mathbb{N} \times \mathbb{N}$, tout point a exactement quatre voisins, excepté l'origine $(0,0)$ qui n'en a que deux, $(0,1)$ et $(1,0)$, et tous les points sur les demi-droite $(x,0)$ et $(0,y)$,

avec $x, y \geq 1$, qui ont trois voisins.

Soit $\mathbb{B} = \{0, 1\}$ la base pour écrire des entiers. Les mots de $\mathbb{B}^*$ peuvent être représentés dans **l'ordre radix** par un arbre binaire complet, où le *kième* niveau contient tous les mots binaires de longueur k, et l'ordre est donné par le parcours en largeur de l'arbre. Afin de différencier un nombre naturel $x \in \mathbb{N}$ de sa représentation binaire, nous allons écrire $\mathbf{x} \in \mathbb{B}^*$. Les arêtes sont définies inductivement par la règle de réécriture $\mathbf{x} \longrightarrow \mathbf{x}\cdot 0 + \mathbf{x}\cdot 1$, avec comme convention que 0 et 1 sont les étiquettes respectivement des arêtes droites et gauches du nœud ayant la valeur $\mathbf{x}$.

La concaténation est étendue au produit cartésien ainsi : pour

$$(\mathbf{x}, \mathbf{y}) \in \mathbb{B}^* \times \mathbb{B}^*, \textit{ et } (\alpha, \beta) \in \mathbb{B} \times \mathbb{B},$$

on a

$$(\mathbf{x}, \mathbf{y}) \cdot (\alpha, \beta) = (\mathbf{x} \cdot \alpha, \mathbf{y} \cdot \beta).$$

Soient $\mathbf{x}$ et $\mathbf{y}$ deux mots binaires ayant la même longueur. Alors la règle

$$(\mathbf{x}, \mathbf{y}) \longrightarrow (\mathbf{x}\cdot 0, \mathbf{y}\cdot 0) + (\mathbf{x}\cdot 0, \mathbf{y}\cdot 1) + (\mathbf{x}\cdot 1, \mathbf{y}\cdot 0) + (\mathbf{x}\cdot 1, \mathbf{y}\cdot 1)$$

définit un graphe $G = (N, R)$ tel que :

(i) La racine est étiquetée $(0, 0)$,

(ii) Chaque nœud de N (excepté la racine) a quatre fils,

(iii) Si un nœud est étiqueté $(\mathbf{x}, \mathbf{y})$, alors $|\mathbf{x}| = |\mathbf{y}|$,

(iv) Les arêtes de R sont non-orientées, c'est à dire qu'elles peuvent être suivies dans les deux directions.

Par convention, les arêtes menant aux fils sont étiquetées par des paires de l'ensemble ordonné $\{(0,0), (0,1), (1,0), (1,1)\}$. Ces étiquettes équipent l'arbre quaternaire d'une structure d'arbre radix telle que l'équation précédente implique que (x', y') est un fils de (x, y) si et seulement si

$$(x', y') = (2x + \alpha, 2y + \beta),$$

pour (x, y) et $(x', y') \in \mathbb{N} \times \mathbb{N}$ $(\alpha, \beta) \in \{0, 1\} \times \{0, 1\}$. On constate que toute paire d'entiers positifs est représentée exactement une fois dans cet arbre. En effet, si $|\mathbf{x}| = |\mathbf{y}|$ (en ajoutant des zéros à gauche pour le plus petit des deux), la séquence des paires de chiffres (les deux chiffres en première position, les deux chiffres en seconde position, et ainsi de suite) donne un chemin unique dans l'arbre menant à cette paire. Évidemment, la racine ne peut avoir que 3 fils, vu qu'il n'y a pas d'arête étiquetée par $(0, 0)$ commençant à la racine.

On superpose à G la **relation de voisinage** T définie comme suit. Pour toute translation élémentaire $\epsilon \in \{(0, 1), (1, 0), (0, -1), (-1, 0)\}$. Chaque nœud $\textcircled{z} = (x, y)$ est lié à son ϵ-voisin $\textcircled{z} + \epsilon$, quand il existe. Ce qui nous donne un graphe $G' = (N, R, T)$. Si un sous-ensemble donné $M \subset \mathbb{N} \times \mathbb{N}$ doit être représenté, on peut retirer les nœuds non essentiels et le graphe associé ne n'est pas forcément complet.

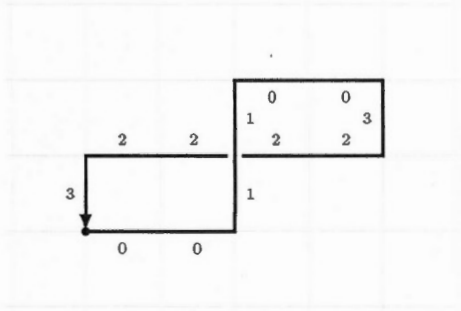


Figure 3.3: Un chemin discret

Exemple 3.2.1. *La figure 3.4 est l'arbre quaternaire associé à la figure 3.3, codée par le mot* $w = 001100322223$*, et qu'on a translatée sur l'origine. Le lien de filiation est représenté par les arêtes en noir tandis que les liens de voisinage sont représentés en rouge. Les nœuds visités sont encadrés en rouge.*

Ainsi, dans l'algorithme de (Brlek, Koskas et Provençal, 2011), on construit l'arbre et le graphe de voisinage en parcourant le mot, tout en notant les nœuds que l'on a visités.

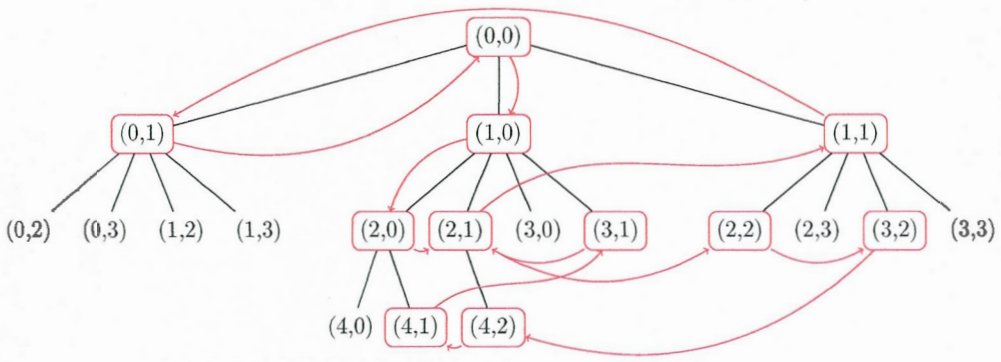


Figure 3.4: Arbre quaternaire associé au mot 001100322223

Il est donc possible de détecter une intersection si l'on visite un nœud déjà visité précédemment. Dans notre algorithme, au lieu de s'arrêter lorsqu'on détecte une intersection, nous construisons l'arbre jusqu'à la fin du mot, afin de pouvoir le parcourir à nouveau.

Soit ⓩ un nœud de G. Son père est noté $f(ⓩ)$, et si ⓩ est étiqueté (x, y), on écrira aussi $f(x, y)$ ou bien $f(\mathbf{x}, \mathbf{y})$. Par convention, la racine ⓡ de G est son propre père. En effet, la racine n'a pas de fils étiqueté par $(0, 0)$ car cela renverrait au même point, donc à la racine elle-même. Si deux nœuds sont voisins, ils ont soit le même père, soir leur père respectif sont voisins. Ainsi, le lemme suivant permet de calculer le père d'un voisin d'un nœud.

Lemme 3.2.2. *Soit $G^{(k)}$ le graphe complet représentant $\mathbb{B}^{\leq k} \times \mathbb{B}^{\leq k}$ ⓩ $= (x, y)$ un nœud de N^k, $\epsilon \in \mathcal{F}$. Si une de ces 4 conditions est valide*

- *$\epsilon = 0$ et $x[k] = 0$,*
- *$\epsilon = 1$ et $y[k] = 0$,*
- *$\epsilon = 2$ et $x[k] = 1$,*
- *$\epsilon = 3$ et $y[k] = 3$,*

alors $f(ⓩ + \epsilon) = f(ⓩ)$. Sinon, $f(ⓩ + \epsilon) = f(ⓩ) + \epsilon$.

Supposons à présent que le nœud ⓧ existe et que son ϵ-voisin ⓨ n'existe pas. La translation $x + \vec{\epsilon}$ est obtenu en trois étapes :

1. Prendre l'arête dans R de $f(\textcircled{x})$ et soit α l'étiquette de cette arête ;
2. Prendre (ou créer) l'arête de T qui va de $f(\textcircled{x})$ vers $\textcircled{z} = f(\textcircled{x}) + \vec{\epsilon}$;
3. Prendre (ou créer) l'arête de R qui va de $\textcircled{z}$ vers $\textcircled{y}$ étiquetée α'

Grâce au lemme 3.2.2, on a $\mathbf{z} \cdot \alpha' = \mathbf{y}$, tel qu'il ne reste qu'à ajouter le lien de voisinage $\textcircled{x} \overset{\epsilon}{\dashrightarrow} \textcircled{y}$. Ainsi, un mot w non vide peut être lu pour construire le graphe G_w.

En commençant par le graphe vide G_ϵ, l'algorithme 5 lis lettre par lettre le mot $w \in \mathcal{F}^*$, construit dynamiquement le graphe G_w en notant les nœuds correspondant comme visité. L'algorithme original s'arrêtait quand un nœud était visité deux fois. Ici, il s'arrête quand le mot est fini, et renvoie le graphe G_w. On rappelle que le graphe vide $G_\epsilon(N_\epsilon, R_\epsilon, T_\epsilon)$ est tel que N_ϵ ne contient que la racine, R_ϵ ne contient que l'arrête allant de la racine vers elle-même, et T_ϵ est vide.

Algorithm 5 ConstruitArbre

Entrée: Un mot $w \in \mathcal{F}^*$ codant un chemin discret.

Sortie: L'arbre G correspondant au mot w.

1: $G \leftarrow G_\epsilon$
2: $\textcircled{c} \leftarrow$ racine de G
3: **pour** i de 1 à $|w|$ **faire**
4: $\quad \epsilon \leftarrow w_i$
5: $\quad \textcircled{z} \leftarrow voisin(G, \textcircled{c}, \epsilon)$
6: $\quad$ **si** $\textcircled{c}$ n'est pas visité **alors**
7: $\quad\quad \textcircled{c}$ est visité
8: $\quad$ **fin si**
9: **fin pour**
10: **renvoyer** G

L'algorithme 6 cherche et créé si nécessaire le ϵ-voisin d'un nœud donné. Le lemme 3.2.2 permet d'effectuer le test de la ligne 6 de l'algorithme 6 en temps constant. En effet, cette condition est effectuée en ne testant que le bit de poids le plus faible d'une seule des coordonnées de la représentation en binaire $\textcircled{c}$. La complexité en temps de cet algorithme

Algorithm 6 voisin

Entrée: Un graphe $G = (N, R, T)$, Ⓒ $\in N$, $\epsilon \in \mathcal{F}$

Sortie: Un nœud Ⓩ ϵ-voisin de Ⓒ

si le lien Ⓒ $\xrightarrow{\epsilon}$ Ⓩ n'existe pas **alors**

Ⓟ $\leftarrow f(\text{Ⓒ})$

si $f(\text{Ⓒ} + \epsilon) =$ Ⓟ **alors**

Ⓡ $\leftarrow$ Ⓟ

sinonⓇ $\leftarrow$ *voisin*$(G, \text{Ⓟ}, \epsilon)$

fin si

Ⓩ $\leftarrow$ fils de Ⓡ correspondant à Ⓒ $+ \epsilon$

Ajouter le lien Ⓒ $\xrightarrow{\epsilon}$ Ⓩ

fin si

renvoyer Ⓩ

est donc entièrement déterminée par l'appel récursif à l'étape 5 de l'algorithme 5. Chaque appel récursif de l'algorithme 5 ajoute un lien de voisinage. Par conséquent, si pour un nœud Ⓩ $\in N$, tous les liens de voisinages ont été ajoutés, il n'y aura pas d'autre appel récursif pour celui-ci. Étant donné qu'un nœud n'a pas plus de 4 voisins, cela borne le nombre d'appel sur un nœud donné à 8. Il nous reste encore à prouver que le nombre de nœuds dans le graphe est proportionnel à $|w|$.

En premier lieu, considérons les nœuds visités. Pour toute lettre lue, au plus un nœud sera noté visité. Il y aura donc au plus $|w|$ nœuds visités. Afin de borner le nombre de nœuds non visités, il faut un lemme technique. La fonctions père $f : N \setminus \{(0,0)\} \mapsto N$ peut être étendue à un sous-ensemble de nœuds de N comme suit : pour $M \subseteq N$, l'ensemble des pères de M est $f(M \{f(\text{Ⓢ}) \mid \text{Ⓢ} \in M\}$. De plus, f peut être itéré pour avoir $f^i(M)$, l'ensemble des ancêtres de rang h d'un ensemble de nœuds M. La fonction f est une **contraction** : $|f(M)| \leq |M|$, et il y a un ancêtre unique pour tous les nœuds, la racine.

Lemme 3.2.3. *Soit* $M = \{n_1, n_2, n_3, n_4, n_5\} \subset N$ *un ensemble de* 5 *nœuds tels que*

$(n_i, n_{i+1}) \in T$ *pour* $i = 1, 2, 3, 4$. *Alors* $|f(M)| \leq 4$.

Cela permet de borner le nombre de nœuds en utilisant le fait que tous les nœuds non-visités sont des ancêtres de nœuds visités, l'exception étant les nœuds non-visités $(0, 1)$ et $(1, 0)$ qui sont créés comme des feuilles à l'étape d'initialisation.

Lemme 3.2.4. *Étant donné un mot* $w \in \mathcal{F}^n$ *et le graphe* $G_w = (N, R, T)$, *le nombre de nœuds de* G_w *est dans* $\mathcal{O}(n)$.

Démonstration. Soit $N_v \subseteq N$ l'ensemble des nœuds visités, et soit h la hauteur de l'arbre (N, R). On a $N = \bigcup_{0 \leq i \leq h} f^i(N_v)$, et donc

$$|N| \leq \sum_{0 \leq i \leq h} |f^i(N_v)|$$

Par construction, l'ensemble N_v forme une séquence de nœuds tels que deux nœuds consécutifs sont voisins. En effet, ces nœuds sont créés en lisant séquentiellement le mot w. Alors, en divisant cette séquence en blocs de longueur 5, on peut appliquer le lemme 3.2.3, et on obtient :

$$|f(N_v)| \leq 4 \left\lceil \frac{|N_v|}{5} \right\rceil \leq \frac{4}{5}(|N_v| + 4).$$

Ce n'est pas le cas pour un chemin qui n'est pas auto-évitant, mais vu qu'un chemin qui s'intersecte aura nécessairement moins de nœuds visités qu'un chemin qui ne s'intersecte pas de même longueur, la borne reste valable. Comme deux nœuds voisins soit partagent le même père, soit ont des pères voisins, il en est de même pour les ensembles $f(N_v)$, $f^2(N_v)$, ..., $f^h(N_v)$. Alors, en combinant les deux inégalités précédentes, on obtient la borne suivante.

$$|N| \leq \sum_{0 \leq i \leq h} |f^i(N_v)| \leq \sum_{0 \leq i \leq h} \left(\left(\frac{4}{5}\right)^i |N_v| + \sum_{0 \leq j \leq h} \left(\frac{4}{5}\right)^j 4 \right)$$

$$\leq |N_v| \left(\frac{1}{1 - \frac{4}{5}}\right) + 4 \sum_{0 \leq i \leq h} \left(\frac{1}{1 - \frac{4}{5}}\right) \leq 5|N_v| + 20h.$$

Étant donné que la hauteur h de l'arbre (N, R) correspond au nombre de bits nécessaires pour écrire les coordonnées des nœuds de N, $h \in \mathcal{O}(\log n)$, et donc $|N| \in \mathcal{O}(n)$. ■

La structure d'arbre quaternaire est donc construite linéairement en temps et en espace. Il est également possible de généraliser l'arbre quaternaire pour un chemin qui ne resterai pas dans le premier quadrant, en effectuant simplement une translation de celui-ci. En effet, étant donné que l'on sauvegarde les coordonnées extrêmes de notre chemin, on peut s'en servir pour calculer les coordonnées de la racine de l'arbre (x_r, y_r) où x_r est la coordonnée x du point W et y_r est la coordonnée y du point S.

3.3 Formalisation

L'algorithme 7 calcule l'enveloppe externe d'un chemin discret w. On commence par construire l'arbre quaternaire G associé à w. Puis, en commençant par le point W de la boîte englobante, on lit le mot w. Aux intersections, pour chaque voisin v de la coordonnée courante c, on calcule la lettre associée au vecteur $\overrightarrow{v} - \overrightarrow{c}$, $Step(\overrightarrow{v} - \overrightarrow{c})$. Puis, on calcule le mot de première différence $\Delta(w_c \cdot Step(v - c))$ du mot formé par la lettre courante w_c de w concaténée à $Step(v - c)$. Ce mot de première différence donnera le virage associé au voisin v. Enfin, on choisira le voisin dont le mot de première différence sera égal à, dans cet ordre de priorité : 3 (virage à droite), 0 (un pas en avant), 1 (un virage à gauche) et 2 (un pas en arrière). La procédure finit quand on retourne au point W et qu'on a exploré tous ses voisins.

Théorème 3.3.1. *Pour tout mot $w \in \mathcal{F}^*$, l'algorithme 7 renvoie* Hull(w).

Démonstration. Soit Hull(w) de longueur $k \in \mathbb{N}^+$. Notre invariant de boucle est :

> *Au début de la ième itération de la boucle while à la ligne 6, w' est une préfixe de longueur i de* Hull(w).

L'invariant est vrai la première fois que la ligne 6 est exécutée, étant donné qu'à ce moment, w' est le premier pas de w calculé à la ligne 5. À présent supposons que l'invariant est vrai avant la $i^{ème}$ itération de la boucle. Alors, les lignes 8 à 13 trouvent le virage le plus à droite à la coordonnée courante `current`. Alors, à la ligne 14, w' est concaténé avec le pas de ce virage. Par la règle de la main droite pour résoudre les labyrinthes connexes simples, considérer le virage le plus à droite donnera les coordonnées

sur l'enveloppe externe de w. Par conséquent, à la fin de l'itération, w' est un préfixe de Hull(w) de longueur $i+1$. Enfin, à la fin de la boucle, w' est un préfixe de Hull(w) de longueur k, et donc $w' = \text{Hull}(w)$. Notons que comme tous les voisins de W sont sur Hull(w), la ligne 15 va enlever efficacement tout élément de N qui deviendra, au final, l'ensemble vide. ■

Algorithm 7 Outer hull

Entrée: Un mot $w \in \mathcal{F}^*$ codant un chemin discret

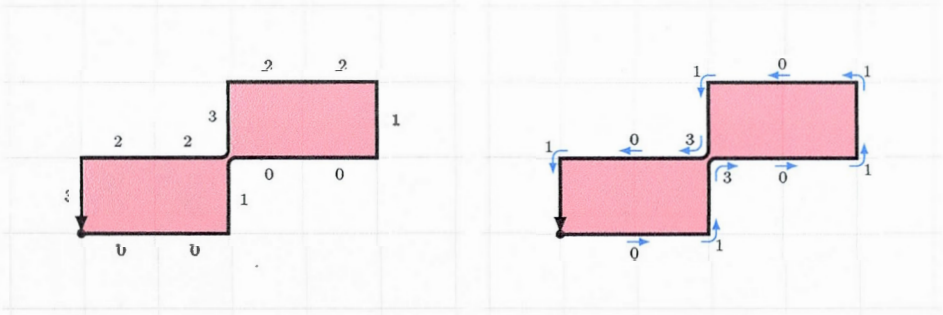
Sortie: Un mot simple $w' \in \mathcal{F}^*$ décrivant Hull(w)

1: Construire l'arbre quaternaire G associé à w enraciné en W
2: Soit W la coordonnée la plus en bas à gauche de la boîte englobante de w
3: Soit N l'ensemble de tous les voisins visités de W
4: $c \leftarrow W + (1,0)$ s'il existe, ou $W + (0,1)$ sinon
5: $w' = \mathit{Step}(c - W)$
6: **tant que** $c \neq W$ et $N \neq \varnothing$ **faire**
7: `temp_turn` $= 2 \mod 4$
8: **pour** chaque voisin v de c **faire**
9: **si** $(\mathit{Step}(v-c) - w'.\mathit{last} + 1) \mod 4 \leq (\texttt{temp\_turn} + 1) \mod 4$ **alors**
10: `temp_turn` $\leftarrow \mathit{Step}(v-c) - w'.\mathit{last}$
11: `next` $\leftarrow v$
12: **fin si**
13: **fin pour**
14: $w' = w' \cdot \mathit{Step}(\texttt{next} - c)$
15: Retirer c de N
16: $c \leftarrow \mathit{next}$
17: **fin tant que**
18: **renvoyer** w'

Montrons maintenant que l'algorithme 7 est linéaire en temps et en espace. En premier lieu, la structure d'arbre quaternaire est construite en temps linéaire (Brlek, Koskas et Provençal, 2011). De plus, comme dit dans la section 1.3, la coordonnée W est facilement

calculée en temps linéaire également. Par conséquent, les calculs des lignes 1 et 2 sont effectués en temps linéaire. Ensuite, les lignes 3, 4 et 5 sont effectuées en temps constant. De plus, l'ensemble N est construit en temps linéaire en accédant aux informations de la racine dans l'arbre quaternaire, donc la ligne 3 est faite en temps linéaire. Ensuite, étant donné que chaque coordonnée a au plus quatre voisins, la boucle for de la ligne 8 est exécutée au plus quatre fois par itération de la boucle while. La ligne 15 s'effectue en temps constant, dû au fait que N a au plus deux éléments. Comme les instructions des lignes 7, 9, 10, 11, 14 et 16 sont toutes calculés en temps constant, au plus $k(4c_1 + c_2)$ calculs sont effectués pendant l'exécution de la boucle while, où $k \in \mathbb{N}^+$ est la longueur de $\text{Hull}(w)$ et $c_1, c_2 \in \mathbb{R}$ sont des constantes, ce qui implique que l'algorithme 7 est linéaire en temps. Ensuite, il découle du fait que la structure d'arbre quaternaire occupe un espace linéaire que l'algorithme est également linéaire en espace.

Exemple 3.3.2. *Considérons le mot de contour* $w = 001100322223$ *de la figure 1.3. Alors, l'algorithme 7 renvoie le mot* $w' = 001001223223$ *(voir la figure 3.5). On peut voir facilement que* w' *est un chemin simple et fermé décrivant l'enveloppe externe de* w. *Donc* $\text{Hull}(w) = w'$.



(a) (b)

Figure 3.5: (a) Enveloppe externe d'un mot (b) sa suite de virages

3.4 Enveloppe convexe de chemins discrets

Montrons à présent comment notre algorithme peut être utilisé pour calculer linéairement l'enveloppe convexe de tout chemin discret.

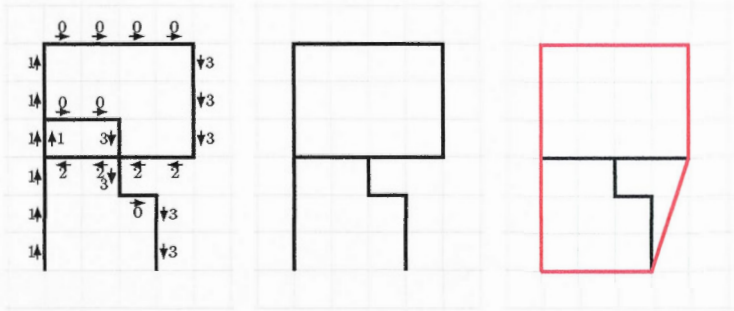
Proposition 3.4.1. *Soit $w \in \mathcal{F}^*$ le mot de contour d'un chemin discret. Alors*

$$\text{Conv}(w) = \text{Conv}(\text{Hull}(w))$$

Démonstration. Il y a deux cas possibles : soit w est déjà simple et fermé, et donc $\text{Hull}(w) = w$, soit w n'est pas simple. Alors, par définition, $\text{Hull}(w)$ est le bord de w. Étant donné que $\text{Conv}(w)$ est la figure convexe d'aire minimale contenant w, elle doit également contenir $\text{Hull}(w)$, et donc $\text{Conv}(w) = \text{Conv}(\text{Hull}(w))$. □

Corollaire 3.4.2. *Il est possible de calculer l'enveloppe convexe de n'importe quel chemin discret*

En effet, de par le fait que $\text{Hull}(w)$ est simple pour tout chemin w, et suite au résultat ci-dessus, la procédure pour calculer l'enveloppe convexe d'un chemin discret est extrêmement simple : Tout d'abord, nous calculons sont enveloppe externe $\text{Hull}(w) = w'$, puis nous calculons l'enveloppe convexe $\text{Conv}(w')$. La première étape se fait évidemment avec l'algorithme 7, et la suivante se fait avec l'algorithme 4. Étant donné que les deux algorithmes sont linéaires en espace et en temps, le calcul de l'enveloppe convexe d'une figure discrète se fait en temps linéaire.



(a) (b) (c)

Figure 3.6: (a) Un chemin, (b) son enveloppe externe et (c) son enveloppe convexe

Exemple 3.4.3. *Voyons au travers d'une figure le processus du calcul de l'enveloppe convexe d'une figure discrète non simple et non fermée. En premier lieu, nous avons la*

figure telle quelle, 3.6(a) codée par le mot $w_1 = 11111100003332222100330 33$. *Ensuite, après application de l'algorithme 7, on obtient la figure 3.6(b), codée par le mot* $w_2 = 11100303311210111222333333$. *Or* $|w_2|_1 = 9 = |w_2|_3$ *et* $|w_2|_0 = 4 = |w_2|_2$, *donc la figure représentée par* w_2 *est fermée. Enfin, maintenant que nous disposons d'une figure fermée simple, il est facile d'appliquer l'algorithme 4 de calcul de l'enveloppe convexe.*

[Cette page a été laissée intentionnellement blanche]

CONCLUSION

Dans ce mémoire, nous avons cherché un moyen de généraliser le calcul de l'enveloppe convexe d'une figure discrète. En effet, utiliser l'algorithme de calcul de l'enveloppe convexe d'une figure discrète nécessite qu'elle soit fermée et simple. Pour pallier cette contrainte, nous avons décrit un nouvel algorithme utilisant les outils élaborés de combinatoire des mots appliquée à la géométrie discrète. Notre algorithme calcule un équivalent simple et fermé d'une figure discrète, appelé "enveloppe externe". Ainsi, on peut appliquer l'algorithme de calcul de l'enveloppe externe d'une figure quelconque, puis appliquer celui de l'enveloppe convexe au résultat, donnant ainsi l'enveloppe convexe de la figure d'origine. Nous avons également établi dans ce mémoire que le calcul de l'enveloppe externe d'une figure est linéaire en temps et en espace. Le calcul de l'enveloppe convexe d'une figure discrète simple et fermée étant également linéaire, il en découle que le calcul de l'enveloppe convexe d'une figure quelconque se faisait linéairement en temps et en espace.

Bien que la procédure de calcul de l'enveloppe convexe soit linéaire, elle requiert les calculs préliminaires du calcul de l'enveloppe externe, c'est à dire celui du point de départ et la construction de l'arbre quaternaire associé à la figure, et le déroulement de l'algorithme 7. Ces premiers calculs imposent de parcourir plusieurs fois le mot de contour de la figure avant de pouvoir calculer son enveloppe convexe. Le coût en est linéaire, mais l'optimisation du nombre de passages reste un problème ouvert.

Il serait intéressant de généraliser ce résultat dans des espaces discrets à trois dimensions, correspondant à de l'étude d'ensembles de cubes unitaires dans $\mathbb{R}^3$. En effet, la structure du graphe de voisinage se prête bien au passage aux dimensions supérieures, avec 6 voisins dans le cas de la dimension 3 et l'arbre quaternaire est remplacé par un arbre octal. De plus, les applications du calcul de l'enveloppe externe ne se limitent pas au

calcul de l'enveloppe convexe. En effet, il est possible de l'utiliser pour pour étudier les unions, intersections et différences d'ensembles discrets.

BIBLIOGRAPHIE

Blondin Massé, A. 2012. « À l'intersection de la combinatoire des mots et de la géométrie discrète : Palindromes, symétries et pavages ». Thèse de Doctorat, Université du Québec à Montréal.

Borel, J.-P., et F. Laubie. 1993. « Quelques mots sur la droite projective réelle ». Journal de théorie des nombres de Bordeaux, vol. 5, no. 1, p. 23–51. <http://eudml.org/doc/93576>.

Brlek, S., M. Koskas et X. Provençal. 2011. « A linear time and space algorithm for detecting path intersection ». Theoretical Computer Science, vol. 412, p. 4841–4850.

Brlek, S., G. Labelle et A. Lacasse. 2005. A Note on a Result of Daurat and Nivat. Coll. Felice, C., et A. Restivo, éditeurs, Coll. « Developments in Language Theory ». T. 3572, série Lecture Notes in Computer Science, p. 189–198. Springer Berlin Heidelberg.

Brlek, S., J.-O. Lachaud et X. Provençal. 2008. Combinatorial View of Digital Convexity. Coll. Coeurjolly, D., I. Sivignon, L. Tougne et F. Dupont, éditeurs, Coll. « Discrete Geometry for Computer Imagery ». T. 4992, série Lecture Notes in Computer Science, p. 57–68. Springer Berlin Heidelberg.

Brlek, S., J.-O. Lachaud, X. Provençal et C. Reutenauer. 2009. « Lyndon+Christoffel=digitally convex ». Pattern Recognition, vol. 42, p. 2239–2246.

Brlek, S., et X. Provençal. 2006. An Optimal Algorithm for Detecting Pseudo-squares. Coll. Kuba, A., L. Nyúl et K. Palágyi, éditeurs, Coll. « Discrete Geometry for Computer Imagery ». T. 4245, série Lecture Notes in Computer Science, p. 403–412. Springer Berlin Heidelberg.

Brlek, S., H. Tremblay, J. Tremblay et R. Weber. 2014. « Efficient computation of the outer hull of a discrete path ». In Proc. eighteenth IAPR International Conference on Discrete Geometry for Computer Imagery. Springler LNCS.

Chan, T. M. 1996. « Optimal output-sensitive convex hull algorithms in two and three dimensions ». Discrete & Computational Geometry, vol. 16, p. 361–368.

Chazelle, B. 1993. « An optimal convex hull algorithm in any fixed dimension ». Discrete & Computational Geometry, vol. 10, p. 377–409.

Christoffel, E. 1875. « Observatio arithmetica ». Ann. Math., vol. 6, p. 145 —152.

Cormen, T. H., C. E. Leiserson, R. L. Rivest et C. Stein. 1990. Introduction to algorithms. MIT Press et McGraw-Hill, third édition.

Daurat, A., et M. Nivat. 2005. « Salient and reentrant points of discrete sets ». Discrete Applied Mathematics, vol. 151, p. 106–121.

Duval, J. 1983. « Factorizing words over an ordered alphabet. ». J. Algorithms, vol. 4, no. 4, p. 363 — 381.

Fredricksen, H., et J. Maiorana. 1978. « Necklaces of beads in k colors and k-ary de bruijn sequences ». Discrete Mathematics, vol. 23, no. 3, p. 207 – 210.

Freeman, H. 1961. « On the encoding of arbitrary geometric configurations ». IRE Transactions on Electronic Computers, vol. EC-10, no. 2, p. 260–268.

Goodman, J. E., et J. O'Rourke. 2004. Handbook of discrete and computational geometry. CRC Press, second édition.

Graham, R. A. 1972. « An efficient algorithm for determining the convex hull of a finite planar set ». Information Processing Letters, vol. 1, no. 4, p. 132–133.

Gruber, P. 2007. Convex and Discrete Geometry. Springer-Verlag.

Kelley, J. L. 1955. General topology. Springer-Verlag, second édition.

Kim, M.-A., E.-J. Lee, H.-G. Cho et K.-J. Park. 1997. « A visualization technique for DNA walk plot using k-convex hull ». In Proceedings of the fifth international conference in Central Europe in computer graphics and visualization, p. 212–221, Plzeň , Czech Republic. Západočeská univerzita.

Kim, Y. S. 1992. « Recognition of form features using convex decomposition ». Computer-Aided Design, vol. 24, no. 9, p. 461–476.

Lothaire, M. 1997. Combinatorics on words. Cambridge University Press.

——. 2005. Applied combinatorics on words. Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Melkman, A. A. 1987. « On-line construction of the convex hull of a simple polyline ». Information Processing Letters, vol. 25, no. 1, p. 11–12.

Provençal, X. 2008. « Combinatoire des mots, géométrie discrète et pavages ». Thèse de Doctorat, Université du Québec à Montréal.

Sherali, H., et W. Adams. 1990. « A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems ». SIAM Journal on Discrete Mathematics, vol. 3, no. 3, p. 411–430.

Spitzer, F. 1956. « A combinatorial lemma and its application to probability theory ».

Transactions of the American Mathematical Society, vol. 82, p. 323–339.

Stein, W., *et al.* 2012. Sage Mathematics Software (Version 4.8). The Sage Development Team. `http://www.sagemath.org`.

Vella, A. 2005. « A fundamentally topological perspective on graph theory ». Thèse de Doctorat, University of Waterloo.

Yao, A. C.-C. 1979. « A lower bound to finding the convex hulls ». Thèse de Doctorat, Stanford University.