Représentation par jeu du chaos de séquences d'ADN

Henin Thibaut, ENS Cachan - Symbiose IRISA

20 juin 2006

Résumé

Le jeu du chaos est une technique permettant de dessiner facilement des images fractales. Cette technique a été adaptée aux séquences d'ADN et permet de représenter des génomes par des images, révélant des structures fractales différentes suivant les génomes. Une question fondamentale est de savoir d'où viennent ces structures fractales, et de les caractériser.

Dans ce travail, nous montrons que ces structures sont la conséquence directe de la présence de langages réguliers au sein de la séquence. Nous proposons ensuite une méthode de quantification de cette présence et de caractérisation des structures fractales à l'aide d'automates finis probabilistes. Nous montrons également que l'on peut utiliser ces structures et ces automates pour discriminer des génomes.

Nous ouvrons par cette technique un certain nombre de pistes de représentations de séquences et leurs applications à diverses problématiques.

Table des matières

Introduction

Cela fait maintenant 3 ans que le génome humain est disponible. Lancé au début des années 90, le projet génome a eu pour but le séquençage du génome humain avant la fin 2005 [1]. Regroupant 6 pays (les USA, le Royaume Uni, le Japon, la France, l'Allemagne et la Chine), il s'est achevé en 2003 avec 2 ans d'avance. Les différents centres de séquençage participants s'étant engagés à diffuser publiquement leurs résultats, le génome humain est disponible dans diverses bases de données publiques sur internet. Cette disponibilité publique du génome a permis entre autre d'éviter une appropriation du génome par une base de donnée privée (et donc un accès au prix fort), mais aussi de faciliter la recherche en génomique.

Malgré cette disponibilité, nous n'en savons toujours pas beaucoup sur la syntaxe et la sémantique de l'ADN. On pense que la séquence d'ADN présente de l'ordre à longue portée [2, 3]. Cet ordre se présente sous plusieurs formes ; comme des similarités [4], des répétitions périodiques [5] ou d'autres façons non encore répertoriées. Il n'est cependant pas toujours évident de caractériser cet ordre (trouver la syntaxe de la séquence) et de lui donner une interprétation biologique (en donner la sémantique). Des grammaires rudimentaires existent pour décrire des séquences d'ARN [6, 7], mais ne sont pas directement applicables aux séquences d'ADN.

Dans [8], il a été suggéré d'utiliser le jeu du chaos pour représenter des séquences d'ADN comme outil dans la recherche de l'ordre à longue portée. Le jeu du chaos est un algorithme permettant de dessiner des images fractales. Appliqué à l'ADN il génère des motifs qui sont parfois difficile à interpréter.

L'objectif principal de ce stage est de fournir des outils d'aide à l'interprétation et d'analyse des images fractales produites par le jeu du chaos. Dans cette optique nous nous sommes consacrés à répondre aux questions suivantes :

1. Quelle est l'origine de la fractalité de la représentation par le jeu du chaos ?
2. Comment caractériser et interpréter les motifs contenues dans les images ?
3. Quelles sont les applications possibles en génomique et les limites de ces applications du jeu du chaos ?

Ce rapport s'organise autour de ces questions. Le chapitre 1 contient la description du jeu du chaos dans le cas général. Dans le chapitre 2, nous faisons quelques rappels sur la théorie des fractales. Dans le chapitre 3, nous introduisons le jeu du chaos et son utilisation pour représenter des séquences d'ADN.

Dans les chapitres 4 et 5, nous expliquons l'origine des structures fractales observées dans les représentations par jeu du chaos. On propose des grammaires régulières simples pour les produire. La présence de ces motifs dans les représentations des séquences suggère l'existence de règles grammaticales régulières dans la séquence d'ADN, mais n'exclut pas l'existence de règles plus complexes.

Dans le chapitre 6, nous donnons un moyen de caractériser plus finement ces motifs. On donne une interprétation statistique des motifs et on montre comment les modéliser par des automates probabilistes, qui permettent aussi de quantifier leur présence. De plus, nous proposons une technique de filtrage des images pour supprimer certains motifs et ainsi permettre d'en découvrir d'autres.

Dans les chapitres 7 et 8, nous décrivons quelques applications possibles de la représentation par jeu du chaos. Nous commençons par utiliser cette représentation pour discriminer des génomes. Ensuite, nous montrons comment la représentation permet une analyse locale des régions de la séquence d'ADN ainsi qu'une analyse de certaines symétries statistiques au sein de la séquence.

Chapitre 1

Le Jeu du Chaos aléatoire

Le jeu du chaos [8, 9] est un algorithme permettant de dessiner des images fractales en faisant décrire une trajectoire à un point dans le plan. À chaque étape de l'algorithme, on fait bouger un point en lui appliquant une transformation choisie parmi un ensemble fini. Le jeu du chaos se définit par un ensemble de transformations et par une méthode de choix de la transformation à chaque étape.

- Les transformations consistent à déplacer le point à mi-chemin entre lui et un point de contrôle. Ces points de contrôles sont immobiles et permettent de paramétrer l'allure de la figure finale. On définit alors une transformation par point de contrôle.
- Le choix de la transformation équivaut au choix du point cible. Ce choix peut être aléatoire, déterministe ou obéissant à un processus arbitraire.

En faisant varier certains paramètres de l'algorithme (tels que la transformation, le nombre de points cibles, la façon de les choisir, ...) on peut observer différentes structures, plus ou moins fractales comme nous le verrons plus loin.

1.1 Définition formelle du jeu du chaos

A partir des n points de contrôle $p_0, p_1, ...p_n$ de $\mathbb{R}^2$, on définit les transformations $u_i : \mathbb{R}^2 \to \mathbb{R}^2$ par $u_i(y) = (y + p_i)/2$ pour $i \in \{1...n\}$. Pour simplifier les notations, on note $u_{i_1 i_2 ... i_k}$, la fonction $u_{i_k} \circ u_{i_{k-1}} \circ ... \circ u_{i_1}$ (et u_ϵ la fonction identité).

Définition 1.1.1. À partir d'une suite $c = (c_i)_{i \in \mathbb{N}^*} \in \{1...n\}$, on définit la suite des points $(y_k)_{k \in \mathbb{N}}$ par l'itération des transformations sur le point initial y de la manière suivante :

$$\forall y \in \mathbb{R}^2, y_0 = y \text{ et } \forall k \in \mathbb{N}^*, y_k = u_{c_k}(y_{k-1}) = u_{c_1 ... c_k}(y)$$

Cette suite s'exprime de manière non récursive de la façon suivante :

$$\forall y \in \mathbb{R}^2, k \in \mathbb{N}, y_k = \frac{1}{2^k} \left(y + \sum_{i=1}^{k} p_{c_i} 2^{k-1} \right)$$

Définition 1.1.2. On appelle jeu du chaos (CG), l'orbite du point y par le système de transformations décrit plus haut (aussi appelé attracteur du système) :

$$\forall y \in \mathbb{R}, CG(y) = \{y_k : k \in \mathbb{N}\}$$

1.2 Algorithme du jeu du chaos aléatoire

Le jeu du chaos aléatoire est un cas particulier du jeu du chaos puisque le choix de points cibles se fait toujours de manière aléatoire.

L'algorithme du jeu du chaos aléatoire est donné en algo. 1. Il utilise un générateur aléatoire ($random \in [0, 1[$) pour simuler la suite (c_i) des choix de points cibles (*i.e.* de loi uniforme sur $\{1..n\}$ où n est le

Algorithme 1 Algorithme du jeu du chaos classique

Entrées: $k \in \mathbb{N}, P$ tableau de points, y_0 point initial
Sorties: L'ensemble des points à afficher

$y \leftarrow y_0$
$L \leftarrow \{y\}$
$j \leftarrow 0$
tantque $j \neq k$ **faire**
  $i \leftarrow random * |P|$
  $y \leftarrow (y + P_i)/2$
  $L \leftarrow L \cup \{y\}$
  $k \leftarrow k + 1$
fin tantque
Retourner L.

nombre de points) et un entier k qui contrôle le nombre d'itérations dans la boucle (et donc de points à afficher).

Dans l'exemple, donné Fig. 1.2.1, on considère trois points de contrôle : P_1, P_2, P_3 et on fait bouger un point (bleu) en mémorisant sa trajectoire (en noir). Dans l'exemple, le point bleu se déplace successivement vers les points P_1, P_2, P_1 puis P_3.

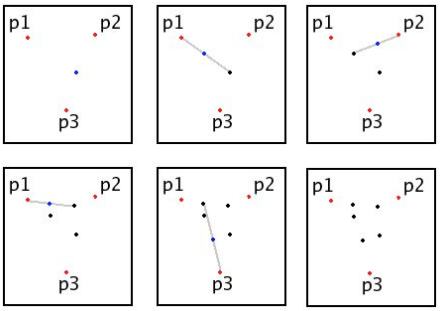

FIG. 1.2.1 – 4 premières étapes d'un jeu du chaos avec trois points de contrôle

1.3 Exemple d'attracteurs

En considérant trois points de contrôle, et en effectuant suffisamment d'itérations, on construit le triangle de Sierpinski [Fig. 1.3.1]. Cet exemple est intéressant car il montre qu'en jouant avec un processus aléatoire (du chaos), on peut produire des images très structurées (et même fractales).Cette figure est autosimilaire puisqu'elle est formée de trois copies plus petites d'elle même.

Néanmoins, on obtient pas toujours de figures ramifiées ; par exemple, en utilisant quatre points de contrôle placés aux sommets d'un carré, le jeu du chaos remplit complètement le carré dont les sommets sont les points de contrôle [Fig. 1.3.2]. Néanmoins, ces structures ramifiées reviennent quand on change la probabilité de choix des points cibles. La figure 1.3.2 a été obtenue en attribuant une probabilité d'être choisi différente à chaque point de contrôle. La probabilité du choix des points, dans le sens anti-horaire et en partant du haut à gauche est de $\frac{1}{10}$, $\frac{2}{10}$, $\frac{3}{10}$ et $\frac{4}{10}$.

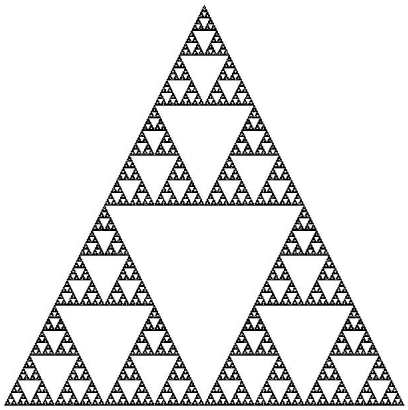

FIG. 1.3.1 – Les 500 000 premières itérations du jeu du chaos aléatoire avec trois points de contrôle produisent le triangle de Sierpinski.

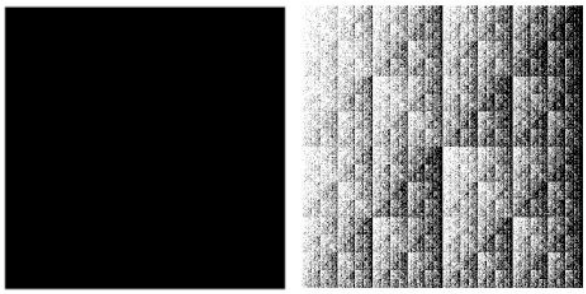

FIG. 1.3.2 – Les 500 000 premières itérations du jeu du chaos aléatoire avec quatre points de contrôle. Un choix des points cible suivant une loi uniforme remplit complètement le carré (à gauche), un choix suivant une probabilité différente par point produit une image plus fractale.

Chapitre 2

Les fractales

Il n'existe pas de définition rigoureuse des fractales mais on retient généralement quelques propriétés courantes pour les définir. Ces deux caractéristiques principales sont les suivantes :

- Le caractère plein/vide des figures fractale nécessite une définition différente de la dimension au sens classique (que nous donnons en section 2.1). La dimension des objets fractals n'est généralement pas un nombre entier.
- Le caractère répétitif des figures fractales se traduit en terme d'auto-similarité est est expliqué en section 2.2.

Le triangle de Sierpinski [FIG. 1.3.1] que nous avons déjà rencontré est un très bon exemple d'objet fractal puisqu'il a un caractère très répétitif et une dimension non entière. Son caractère répétitif vient du fait qu'il est formée de trois copies plus petites de lui-même. Ainsi, on retrouve des reproductions du triangle de Sierpinski un peu partout à l'intérieur de la figure. Il est également difficile de donne une dimension au sens classique du terme. C'est une figure de longeur infinie contenue dans une surface bornée. De plus, sa surface est nulle. Cette figure a une dimension entre 1 et 2 (entre un segment et une surface). Comme nous la calculerons, la dimension du triangle de Sierpinski est de $1,585$.

2.1 Dimension de Hausdorff

Les deux notions principales associées à l'étude des fractales que nous allons introduire ici sont les suivantes : la distance et la dimension de Hausdorff.

La distance de Hausdorff : Cette distance permet de définir une distance entre ensembles de points et est définie par :

$$\forall A, B \subset E, d_H(A,B) = \max_{a\in A}(\min_{b\in B} d(a,b)) \tag{2.1.1}$$

On note que H n'est pas symétrique dans tous les cas. Par exemple, la distance d'un point p du plan (placé en $(2,0)$) vers un cercle c (centré en $(0,0)$ et de rayon 1) est différente suivant l'ordre des paramètres ; $H(p,c) = 1$ tandis que $H(c,p) = 3$. Pour obtenir une distance, on définit $d(A,B) = max(d_H(A,B), d_H(B,A))$; qui est symétrique et définit une métrique. Cependant, on se contente parfois de d_H car elle donne déjà suffisamment d'informations à elle seule.

La dimension de Hausdorff : Cette dimension est une valeur permettant de comparer une figure fractale à des ensembles géométriques traditionnels. Cette valeur est entière pour toutes les formes géométriques classiques (cercles, carrés, cubes, ...) mais ne l'est pas pour les objets fractals (triangle de Sierpinski, ...). Pour un ensemble $E \subset \mathbb{R}^n$, sa dimension de Hausdorff est comprise entre k et $k+1$ si E contient des boules de $\mathbb{R}^k$ mais pas de $\mathbb{R}^{k+1}$. On définit la dimension de Hausdorff $dim_{\mathcal{H}}E$ de la manière suivante :

$$\mathcal{H}^s_\delta(E) = inf\{\sum_{i\geq 0} |U_i|^s : U_i \text{ recouvrement dénombrable de } E \wedge 0 \leq |U_i| \leq \delta\} \tag{2.1.2}$$

$$\mathcal{H}^s = \lim_{\delta \to 0} \mathcal{H}^s_\delta \tag{2.1.3}$$

$$dim_{\mathcal{H}}(E) = d : \forall s, \mathcal{H}^s(E) = \begin{cases} 0 & \text{si } s < d \\ \infty & \text{si } s > d \end{cases} \tag{2.1.4}$$

Voici quelques exemples d'ensembles et leurs dimensions de Hausdorff respectives (la dimension du triangle de Sierpinski et de la courbe de VonKoch seront calculée plus loin) :

Point	0
Droite	1
Boule de $\mathbb{R}^n$	n
Courbe de Koch	$\log 4/\log 3 = 1.261$
Triangle de Sierpinski	$\log 3/\log 2 = 1.585$

2.2 Les IFS pour construire des ensembles autosimilaires

Définition 2.2.1. Un ensemble est auto-similaire s'il est construit à partir de copies plus ou moins déformées de lui-même. Plus formellement, on définit un ensemble auto-similaire comme un ensemble E tel qu'il existe des transformations $\{T_i\}_{i\in I}$ tels que $E = \bigcup_{i\in I} T_i(E)$

Définition 2.2.2. Les IFS [10] sont des systèmes de transformations $\{T_i\}_{i\in I}$ appliquées itérativement. Ainsi, on définit la suite E_k des étapes de la transformation telle que pour une étape k, $E_k = \bigcup_{i\in I} T_i(E_{k-1})$. On appelle attracteur de l'IFS l'ensemble E, s'il est point fixe de la transformation $E = T(E) = \bigcup_{i\in I} T_i(E)$. Par définition, cet ensemble, quand il existe, est auto-similaire.

Proposition 2.2.3. *Si les transformations T_i de l'IFS sont des similarités*[1] *de ratio c_i, la dimension de Hausdorff de l'attracteur de cet IFS est le réel s satisfaisant l'équation suivante[10] :*

$$\sum_{i=1}^{k} c_i^s = 1 \tag{2.2.1}$$

On généralise la notion d'IFS à des séquences d'objets grâce aux L-systèmes (système de Lindenmayer)[11]. Ces systèmes consistent en un ensemble de symboles (associés à des objets graphiques) et une transformation f remplaçant les symboles par des listes d'autres symboles. La représentation de la limite de tel processus, quand elle existe, est une image fractale.

Définition 2.2.4. On définit un L-système d'après les quatre éléments suivants :

1. l'alphabet Σ est l'ensemble des symboles,
2. l'interprétation graphique qui permet d'associer une action graphique (bouger, tracer un cercle de rayon α, tracer un segment, tourner d'un angle r, ...) à chaque lettre de Σ,
3. l'axiome A est un mot de Σ^* permettant d'initialiser la suite des étapes,
4. la règle de réécriture, $\delta : \Sigma \to \Sigma^*$ qui réécrit les symboles par des listes de symboles.

La suite des étapes $(E_i)_{i\geq 0}$ commence par l'axiome ($E_0 = A$) et est définie récursivement telle que pour toute étape $E_i = e_0.e_1...e_k$, $E_{i+1} = \delta(e_0).\delta(e_1)...\delta(e_k)$. À chaque étape, la figure est réduite d'un facteur permettant de garder sa représentation dans un espace borné et constant.

On définit l'attracteur d'un L-système comme étant la limite de la suite des représentations des étapes du L-système.

Définition 2.2.5. On définit la représentation d'une étape $E_i = e_0.e_1...e_k$ comme étant la succession des actions graphiques correspondants aux lettres de E_i. Par exemple, si x signifie *"tracer un segment vers l'avant"* et $-$ signifie *"tourner d'un angle de $\pi/2$"*, le mot $x - x - x - x$ permet de tracer un carré.

[1] l'application T est une similarité s'il existe un ratio c tel que $0 < c < 1$ et $|T(x) - T(y)| = c|x - y|$ pour tout x et y [10]

2.3 Exemples d'IFS classiques :

La courbe de Koch : C'est une des nombreuses figures obtenues par IFS et L-Système.

L'IFS correspondant est construit à partir de quatre transformations envoyant un segment sur quatre autres segments (réduit d'un facteur d'un tiers)[Fig. 2.3.1].

Il s'agit également d'un L-Système dont les symboles sont les suivants (avec leur interprétation) : x (tracer un segment), $+$ (tourner de $\pi/3$) et $-$ (tourner de $-\pi/3$). La transformation ne modifie que x qui se réécrit $x+x--x+x$ (cette règle est également visualisée dans la fig. 2.3.1). L'axiome est le symbole x. En itérant le processus de transformation [Fig. 2.3.1], on produit une courbe fractale.

On calcule alors sa dimension s d'après l'équation 2.2.1 :

$$\sum_{i=1}^{4} \frac{1}{3}^{s} = 1$$
$$(1/3)^s = 1/4$$
$$s = \log(1/4)/\log(1/3)$$

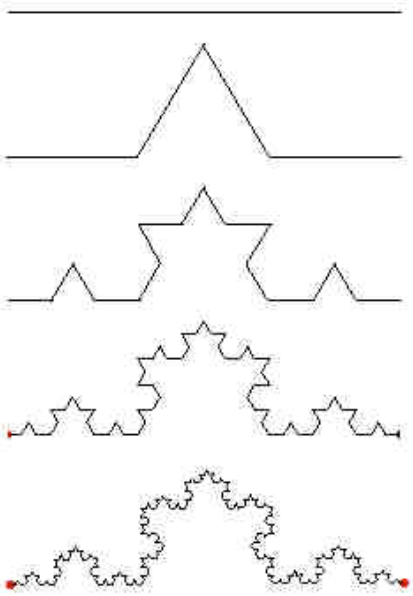

FIG. 2.3.1 – Les quatre premières étapes du processus de calcul de la courbe de Von Koch

Le triangle de Sierpinski : Encore une fois, cette figure est obtenue par IFS et par L-Système.

L'IFS de cette figure est constitué de trois similarités de ratio 1/2 envoyant le triangle sur ses trois extrémités. Par un calcul similaire à celui de la courbe de Von Koch, on trouve $s = \log 3/\log 2$.

Le L-système associé contient les mêmes symboles que pour la courbe de Von Koch ($x, +$ et $-$) plus y (dont la représentation est la même que x, un segment). L'axiome est le symbole x. La transformation ne touche encore une fois que les segments (x et y) et les réécrit d'après les deux règles suivantes (dont les six premières étapes sont dessinées en figure 2.3.2) :

$$x \rightarrow +x-y-x+$$
$$y \rightarrow -y+x+y-$$

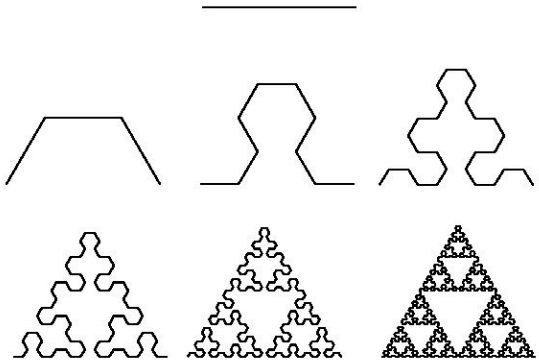

FIG. 2.3.2 – Les six premières étapes du processus de calcul du triangle de Sierpinski

Chapitre 3

Représentation par jeu du chaos des langages sur un alphabet de 4 lettres

Dans [8], il a été suggéré d'utiliser des suites de symboles pour remplacer le générateur aléatoire. L'idée est d'assigner un symbole (ou plusieurs) à chaque point de contrôle. Ensuite, à chaque étape de l'algorithme, on lit une lettre de la suite, qui déterminera le point cible de la transformation. La figure obtenue après la lecture de la suite caractériserait ainsi la suite elle-même.

Dans cette partie, nous allons donner quelques notions en théorie des langages pour permettre de définir formellement la représentation par jeu du chaos de langages puis de séquences. Enfin, on présentera quelques exemples de représentation.

3.1 Théorie des langages

Nous allons définir ici un certain nombre de notions en théorie des langages que nous utiliserons par la suite.

L'alphabet : on note Σ, l'ensemble des lettres de l'alphabet qu'on considère. Dans notre cas, on choisit $\Sigma = \{a, c, g, t\}$, où chaque lettre correspond à une des bases de l'ADN.

Les mots : ce sont des suites finies de lettres de l'alphabet. On définit la longueur d'un mot ω comme le nombre de lettres dans le mot et on la note $|\omega|$. Le mot vide (noté ϵ) est le mot de taille nulle (il ne contient aucune lettre). On note Σ^* l'ensemble des mots de longueur finie. On note aussi Σ^n, l'ensemble des mots de longueur n. Pour un mot u, on définit u_i comme étant la i^{eme} lettre de u. Ainsi, une séquence finie d'ADN peut être vue comme un mot, élément de Σ^*.

La concaténation : étant donnés deux mots (ω_1 et ω_2), leur concaténation est le mot construit avec les lettres de ω_1 suivie de celles de ω_2 (tout en conservant l'ordre des lettres) et on note cette operation : $\omega_1.\omega_2$. Le mot ϵ est élément neutre, autrement dit, on a $\omega.\epsilon = \omega = \epsilon.\omega$. On sait que $|\omega_1.\omega_2| = |\omega_1| + |\omega_2|$. Par contre, cette opération n'est pas commutative. On généralise la concaténation entre mots aux ensembles de mots ; on a ainsi $\omega.E = \{\omega.\mu : \mu \in E\}$ où E est un ensemble de mots (et de même entre deux ensembles de mots).

Les préfixes, suffixes et langages : à partir d'un mot (w), on considère l'ensemble des mots obtenus en retirant la fin de w. On définit ainsi un ensemble $pre(w) = \{u \in \Sigma^* : \exists v \in \Sigma^* : w = u.v\}$ qui contient les préfixes de w. De même, on définit l'ensemble des suffixes de w : $suf(w) = \{u \in \Sigma^* : \exists v \in \Sigma^* : w = v.u\}$. Le langage de w est défini comme l'ensemble des sous-mots de w par $L(w) = pre(suf(w)) = suf(pre(w))$. On note aussi $[w]_n$, l'ensemble des mots de taille n apparaissant dans w ($[w]_n = \Sigma^n \cap L(w)$).

3.2 Définitions formelles de la représentation par jeu du chaos des langages et des séquences

Utiliser le jeu du chaos avec des mots d'un alphabet revient à attribuer un point de contrôle à chaque lettre, d'utiliser les mots à la place de la suite de choix et de commencer par un point précis dans le plan. Dans notre cas, on définit les quatre points de contrôle suivants : $p_a = (0,0)$, $p_c = (0,1)$, $p_g = (1,1)$ et $p_t = (1,0)$. Les transformations sont de la même forme que dans la version aléatoire, mais indexées par les lettres ; elles sont de la forme $u_x(y) = (y + p_x)/2$ pour $x \in \Sigma$. De même, au lieu de la suite d'entiers $(c_i)_{i\in\mathbb{N}}$, on utilise un mot (suite de lettres) $w = w_0...w_k \in \Sigma^*$. Le point initial y est toujours au centre des quatre points de contrôle ($y_0 = (\frac{1}{2}, \frac{1}{2})$). De cette manière, on est sûr que tous les points produits seront dans le carré $]0,1[^2 \subset \mathbb{R}^2$.

Définition 3.2.1. Pour tout mot $w = w_1.w_2...w_k$, les coordonnées d'un point y après avoir lu w sont données par la fonction suivante :

$$\forall y \in]0,1[^2, y_\epsilon = y \text{ et } \forall w \in \Sigma^+, |w| = k, y_w = u_{w_k}(y_{w_1...w_{k-1}}) = (u_{w_k} \circ ... \circ u_{w_1})(y) = u_w(y)$$

Définition 3.2.2. La représentation par jeu du chaos CGR d'un langage L est l'ensemble des points obtenus à partir du point central $(\frac{1}{2}, \frac{1}{2})$ en lisant chaque mot de l. Elle est donnée de la façon suivante :

$$\text{Soit } y = \left(\frac{1}{2}, \frac{1}{2}\right),\ \forall L \subset \Sigma^*, CGR(L) = \{u_w(y) : w \in L\}$$

Définition 3.2.3. La représentation par jeu du chaos CGR d'un mot de Σ^* est définie comme étant la représentation du langage de ses préfixes. On peut aussi la voir comme l'orbite du point en lisant le mot w. Autrement dit, on a la formule suivante :

$$\forall w \in \Sigma^*, CGR(w) = CGR(pre(w))$$

Il s'agit d'un cas particulier de la représentation par jeu du chaos aléatoire et l'algorithme associé [Algo. 2] n'est qu'une variante de l'algo. 1. En effet, on change le générateur aléatoire par une lecture du mot, le nombre d'itérations est contrôlé par la taille du mot et le point initial est fixé.

Algorithme 2 Algorithme du jeu du chaos pour les séquences de Σ^*

Entrées: $w \in \Sigma^*$
Sorties: L'ensemble des points à afficher

```
P ← {a → (0,0), c → (0,1), g → (1,1), t → (1,0)}
y ← (1/2, 1/2)
L ← {y}
k ← 1
tantque k ≤ |w| faire
    i ← w_k
    y ← (y + P_i)/2
    L ← L ∪ {y}
    k ← k + 1
fin tantque
Retourner L.
```

3.3 Exemple de représentation

On peut alors tracer des figures pour représenter des séquences d'ADN [8, 12]. Nous avons ici représenté les séquences du chromosome 14 de l'homme et de E. Coli [Fig. 3.3.1]. Par contre, pour des raisons de visibilité, nous avons restreint le nombre de bases de la séquence à 500 000 (au delà, il y a trop de points et on ne voit plus qu'un carré plein). On peut remplacer l'ensemble par une densité représentée par niveaux de gris (voir section 6.1) ; dans ce cas, on est limité uniquement par la taille du pixel.

Chaque génome produit des images plus ou moins structurées. Une question fondamentale est de pouvoir caractériser et quantifier ces structures. Par exemple, dans la représentation par jeu du chaos du génome humain [Fig. 3.3.1], on distingue plusieurs motifs : le premier (et le plus visible) est le trou, en haut à droite, qui se répète un peu partout en plus petit dans l'image. Ensuite, en regardant bien, on peut voir que les diagonales du carré sont un peu plus marquées. Par contre, sur un génome comme E. Coli [Fig. 3.3.1], on a le sentiment que l'image contient un certain ordre, qu'on ne peut pas le caractériser aussi facilement que pour l'homme.

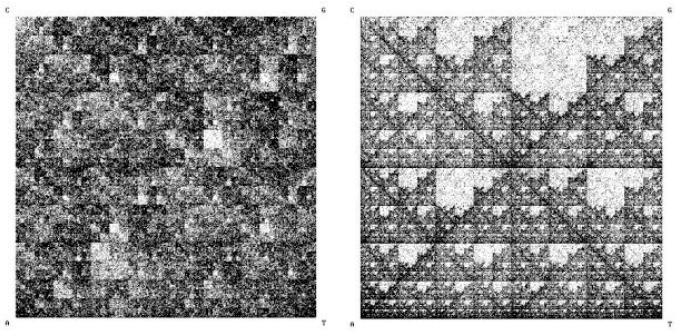

FIG. 3.3.1 – Le jeu du chaos avec les 500 000 premières bases du chromosome 14 de l'Homme (à droite), et de E. Coli (à gauche) (Chaque point noir correspond à une itération de l'algorithme)

Chapitre 4

Auto-similarité de la représentation des langages réguliers à l'aide d'IFS

4.1 IFS générique pour représenter le jeu du chaos

On peut voir les transformations du jeu du chaos comme un ensemble de transformations d'un IFS dont le carré est l'attracteur. Les quatre transformations du jeu du chaos envoient le carré unitaire vers un de ses quartiers. Ainsi, le carré est défini comme un objet auto-similaire, un carré formé par l'union de quatre carrés. De cette façon, on calcule sa dimension de Hausdorff valant 2 (puisque unique solution de $4 * (1/2)^s = 1$).

Définition 4.1.1. On définit le L-Système correspondant au carré construit par l'IFS du jeu du chaos comme il suit.

Ce L-Système contient deux symboles interprétés comme des points (I et F) et quatre symboles (un par lettre de l'alphabet) comme des segments orientés ($\forall x \in \Sigma, L_x$ est orienté vers p_x). L'axiome est I. Se système contient les cinq règles de réécritures suivantes :

$$I \to F\ (L_a I L_g)\ (L_c I L_t)\ (L_g I L_a)\ (L_t I L_c)$$
$$L_a \to L_a L_a | L_c \to L_c L_c | L_g \to L_g L_g | L_t \to L_t L_t$$

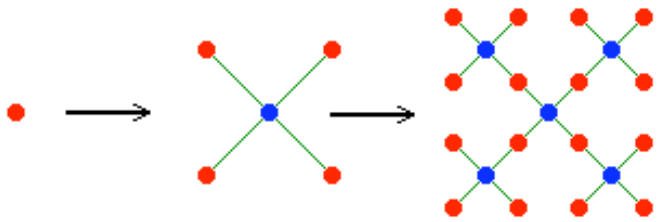

FIG. 4.1.1 – Les deux premières itérations de l'IFS utilisé pour modéliser le jeu du chaos. Le symbole I est représenté par un point rouge et F est représenté par un point bleu. Les segments sont tous représentés en vert.

Nous avons représentés les trois premières étapes de ce L-système en figure 4.1.1. Dans cette figure, le symbole I est représenté par un point rouge et F est représenté par un point bleu. Les segments sont tous représentés en vert. Cette suite de transformations graphiques correspond à la suite des trois premières étapes du L-système :

$$
\begin{aligned}
I \rightarrow & F\ (L_a I L_g)\ (L_c I L_t)\ (L_g I L_a)\ (L_t I L_c) \\
\rightarrow & F \\
& (L_a L_a F\ (L_a I L_g)\ (L_c I L_t)\ (L_g I L_a)\ (L_t I L_c) L_g L_g) \\
& (L_c L_c F\ (L_a I L_g)\ (L_c I L_t)\ (L_g I L_a)\ (L_t I L_c) L_t L_t) \\
& (L_g L_g F\ (L_a I L_g)\ (L_c I L_t)\ (L_g I L_a)\ (L_t I L_c) L_a L_a) \\
& (L_t L_t F\ (L_a I L_g)\ (L_c I L_t)\ (L_g I L_a)\ (L_t I L_c) L_c L_c)
\end{aligned}
\tag{4.1.1}
$$

Proposition 4.1.2. *Chaque point p tracé par ce L-Système correspond au point image d'un mot par le jeu du chaos.*

Démonstration. En associant à chaque mot w de Σ^* deux symboles du L-Système (I_w et F_w) représentés par un point et en utilisant I_ϵ comme axiome, on définit les règles de réécritures de la forme :

$$I_w \rightarrow F_w L_a I_{a.w} L_g L_c I_{c.w} L_t L_g I_{g.w} L_a L_t I_{t.w} L_c$$

Néanmoins, par soucis de clarté, nous n'utiliserons qu'une version simplifiée de la règle ci-dessus : $I_w \rightarrow I_{a.w} I_{c.w} I_{g.w} I_{t.w}$.

Pour coller au jeu du chaos, on trace toujours le point central en $(1/2, 1/2)$ et la longueur des segments est divisées par deux à chaque étape du L-système.

Quel que soit le mot w, le point p_w tracé par les symboles I_w et F_w est sur le point y_w (obtenu par jeu du chaos). □

La représentation de ce L-système correspond à la représentation de Σ^* mais on peut adapter ce L-système pour représenter n'importe quel langage. Le principe est d'ajouter des symboles représentés par des points et de faires des règles de réécritures de ces symboles différentes.

Proposition 4.1.3. *La représentation par jeu du chaos d'un langage est caractérisable par un L-Système dont le nombre de règles et de symboles n'est pas nécessairement fini.*

Démonstration. Il suffit pour cela d'utiliser l'IFS munis des symboles associés aux mots et d'attribuer une représentation différente si le mot du symbole appartient ou non au langage à représenter. Un mot du langage aura ses symboles correspondants affichés comme des points noirs. A l'inverse, les mots en dehors du langage auront des symboles sans représentation graphiques, il ne seront pas affichés.

On peut néanmoins simplifier le L-système en n'exprimant pas de règle pour certains symboles inutiles. Si un mot w n'est suffixe d'aucun mot du langage, quelque soit le mot u qu'on lui concatène en tête $(u.w)$, ces mot n'apparaîtront jamais dans le langage. Les règles de réécriture associées aux symboles de ces mots ($I_{u.w}$) ne produiront que des points non affichés. Il est donc inutile de spécifier ces règles. □

4.2 Calcul des L-systèmes des langages réguliers à l'aide d'automates

Définition 4.2.1. Les langages réguliers sur l'alphabet Σ sont des langages définis récursivement par :

1. le langage vide ($\emptyset$) et les langages $\{x\}, x \in \Sigma$ sont réguliers,
2. L_1 et L_2 sont réguliers $\Rightarrow L_1 + L_2$[1], $L_1.L_2$[2] et L_1^*[3] sont réguliers.

Définition 4.2.2. On définit un automate A comme un quintuplet $(E, \Sigma, I, \delta, F)$ dont les éléments ont la signification suivante :

- E est l'ensemble (fini) des états de l'automate.
- Σ est l'alphabet de l'automate (ensemble fini de lettres).
- $I \subset E$ est l'ensemble des états initiaux.

[1] $L_1 + L_2 = L_1 \cup L_2$
[2] $L_1 + L_2 = \{w\Sigma^* : \exists w_1 \in L_1, w_2 \in L_2, w = w_1.w_2\}$
[3] L_1^* est le plus petit langage contenant les mots de L_1 et ϵ qui soit stable par concaténation.

– $\delta \subset E \times \Sigma \times E$ est l'ensemble des transitions. On note $(e_i, \alpha, e_f) \in \delta$, la transition qui va de e_i vers e_f étiquetée par la lettre α.
– $F \subset E$ est l'ensemble des états terminaux.

Un chemin dans l'automate est une suite $e_1, ...e_k$ d'états de l'automate et on dit que l'automate reconnait un mot $w \in \Sigma^*$ s'il existe un chemin $e_1, ...e_k$ qui débute sur un état initial, passe par des transitions étiquetées par les lettres du mot et termine sur un état final. Autrement dit, $e_1 \in I$, $e_k \in F$ et $\forall i \in \{1..k-1\}, (e_i, w_i, w_i + 1) \in \delta$. Le langage reconnu par l'automate A, noté $L(A)$, est l'ensemble des mots reconnus par A.

Définition 4.2.3. Un automate $A = (E, \Sigma, I, \delta, F)$ est complet et déterministe si, pour chaque état e et chaque lettre α, il existe un et un seul état cible d'une transition débutant sur e étiquetée par α et qu'il n'y a qu'un seul état initial. Formellement, l'automate A est déterministe complet s'il satisfait la propriété suivante :

$$|I| = 1 \ \wedge \ \forall e \in E, \alpha \in \Sigma, |\{s \in E : (e, \alpha, s) \in \delta\}| = 1$$

Il existe un grand nombre de propriétés connues sur les automates et les langages réguliers. Nous ne les détaillerons pas toutes ici, mais citerons les plus intéressantes pour ce travail. Les propriétés utiles sont les suivantes :

1. Les langages reconnus par les automates sont les langages réguliers
2. Pour tout automate A, il existe un automate complet déterministe reconnaissant $L(A)$.
3. Le complémentaire d'un langage régulier est un langage régulier.
4. L'inverse d'un langage régulier[4] est un langage régulier.
5. L'union et l'intersection de langages réguliers sont des langages réguliers.

En figure 4.2.1, nous avons représenté un automate reconnaissant le langage des mots ne contenant pas le doublon "cg". Cet automate à trois états, dont l'état 1 qui est initial et les états 1 et 2 sont terminaux. L'état 3 joue le rôle d'état puits, un état non terminal à partir duquel on ne peut plus sortir.

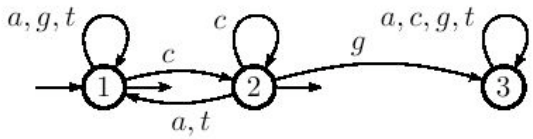

FIG. 4.2.1 – Automate du langage interdisant le sous-mot "cg".

Proposition 4.2.4. *La représentation par jeu du chaos d'un langage régulier est caractérisable par un L-système fini.*

Démonstration. Soit L, le langage considéré, L' le langage inverse de L et l'automate A, reconnaissant le langage L'.

A partir de A, on construit une variante du L-système décrit plus haut de la manière suivante :
– Les symboles : en plus des symboles représentés par des segments $(L_x, x \in \Sigma)$, on définit, pour chaque état e, les symboles I_e et F_e. Leur représentation est un point noir si l'état est terminal, et un point blanc sinon. L'axiome de ce L-système est I_ϵ.
– Les règles de réécritures : en plus des règles appliquées aux segments $(L_x \rightarrow L_x L_x, x \in \Sigma)$, on définit des règles traitant les autres symboles. Pour chaque état de l'automate, on se sert des transitions qui en partent pour construire une règle. Pour un état e, on construit la règle (simplifiée) suivante :

$$I_e \rightarrow F_e I_{\delta(e,a)} I_{\delta(e,c)} I_{\delta(e,g)} I_{\delta(e,t)}$$

où $\delta(e, \alpha)$ est la fonction de transition qui donne la cible de la transition partant de e étiquetée par α.

[4] L'inverse d'un langage L est le langage des mots de L écrits en sens inverse.

Il suffit alors de montrer que pour tout mot w, le point y_w (du jeu du chaos) est tracé par les symboles I_w et F_w correspondants à l'état sur lequel aboutissent les transitions en partant de l'état initial et en lisant w. Dans ce cas, le fait qu'un mot w appartienne au langage sera équivalent au fait que le point correspondant y_w (du jeu du chaos) soit tracé en noir par le L-Système.

On montre cette propriété par récurrence sur la longueur des mots considérés. Pour une longueur nulle, il n'y a qu'un mot (ϵ), dont l'image est le point central, ce qui confirme l'hypothèse.

On suppose donc que cette propriété se vérifie toujours pour les mots de taille n et on pose w de taille $n+1$. On note w', le suffixe de taille n de w et on a $w = \alpha.w'$. Par hypothèse de récurrence, on sait que le point $y_{w'}$ est tracé par le symbole $I_{w'}$ et $F_{w'}$ (qui sont les symboles de l'état $e_{w'}$ de l'automate en ayant lu w').

Par construction du L-système (et par la propriété 4.1.2), on a que $I_{w'} \rightarrow F_{w'} I_{a.w'} I_{c.w'} I_{g.w'} I_{t.w'}$ où les $I_{x.w'}$ sont les états cibles en partant de $e_{w'}$ et en utilisant la transition étiquetée par x. Quel que soit la lettre α en tête de w, le symbole associé au point y_w est bien celui associé à l'état de l'automate en lisant w (par construction du L-Système). □

Ce résultat est important car il montre que les langages réguliers ont une représentation autosimilaire. En effet, c'est le nombre fini de symboles et donc de règles de transformations qui garanti que l'attracteur du système soit auto-similaire. Ainsi, tout langage régulier aura une représentation par jeu du chaos autosimilaire.

On peut alors représenter plus facilement des langages réguliers. Pour représenter le mot où "cg" est interdit comme sous mot, on commence par construire l'automate du langage inverse [Fig. 4.2.2]. Cet automate reconnait le langage où "gc" est interdit comme sous-mot. Le L-système correspondant contient les six symboles suivants : $I_1, F_1, I_2, F_2, I_3, F_3$ et les trois règles (simplifiées) suivantes :

$$I_1 \rightarrow F_1 I_1 I_1 I_2 I_1$$
$$I_2 \rightarrow F_2 I_1 I_3 I_2 I_1$$
$$I_3 \rightarrow F_3 I_3 I_3 I_3 I_3$$

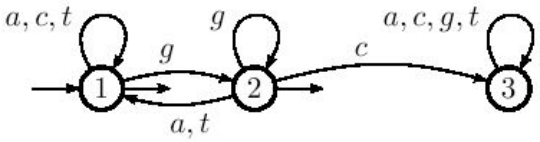

FIG. 4.2.2 – Automate du langage interdisant le sous-mot "gc".

Dans la figure 4.2.3, nous avons représenté les cinq premières étapes du L-système décrit plus haut. Cependant, pour gagner en lisibilité, nous avons choisi d'afficher des carrés (et non des points qui sont trop petits). De plus, nous avons choisi de les colorer en fonction de l'état correspondant aux symboles (rouge pour l'état 1, bleu pour le 2 et jaune pour le 3) et de mettre le tout sur fond noir pour augmenter le contraste et permettre de visualiser l'application des règles.

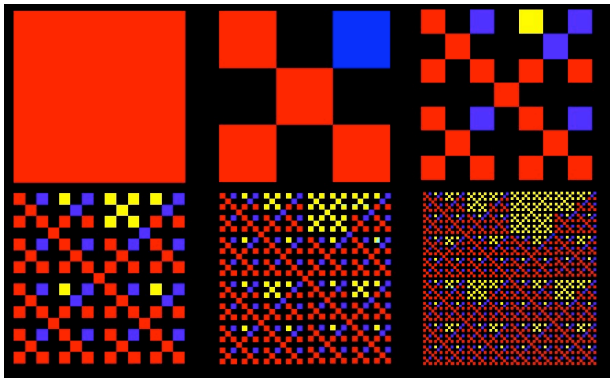

FIG. 4.2.3 – Cinq premières étapes du L-système du langage des mots où "cg" est interdit.

Chapitre 5

Caractérisation des structures fractales dans les représentation des séquences

5.1 Définition d'une distance sur le carré unitaire

5.1.1 Définition des cadrans

La représentation par jeu du chaos permet, entre autre, de regrouper visuellement les images des mots de même suffixe dans des cadrans. On note qu'en ajoutant une lettre de plus, les points des images de n'importe quel mot se retrouvent dans le même quartier de la figure. On continue le raisonnement en constatant qu'en ajoutant une autre lettre, les images des mots se retrouvent encore une fois regroupées dans des carrés plus petit encore [FIG. 5.1.1].

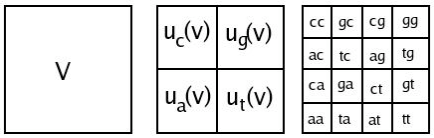

FIG. 5.1.1 – Illustration du découpage de la figure en cadrans regroupant les images de mots dont les suffixes sont communs

Définition 5.1.1. Les cadrans (d'ordre $k \in \mathbb{N}$) sont définis comme des pavés contigus du plan de taille 2^{-k} :

$$\forall k, i, j \in \mathbb{N} : 0 \leq i, j < \ 2^k \ E^k_{i,j} =]2^{-k}.i, 2^{-k}.(i+1)[\times]2^{-k}.j, 2^{-k}.(j+1)[$$

Définition 5.1.2. L'image du carré unitaire par la transformation géométrique associée à un mot est définie de la manière suivante :

$$\forall w \in \Sigma^* \ U_w = u_w(]0,1[^2)$$

Proposition 5.1.3. *Ces deux définitions sont en fait équivalentes et on peut utiliser l'une ou l'autre suivant les besoins sans perdre d'information. En fait on fait correspondre chaque $E^k_{i,j}$ et chaque U_w de la façon suivante (où y_w est défini comme dans la définition 3.2.1) :*

$$\forall w \in \Sigma^* : |w| = k, \ (p_1, p_2) = 2^k y_w - (1/2, 1/2) \implies U_w = E^k_{p_1,p_2}$$

5.1.2 Définition de la distance entre points du carré

Définition 5.1.4. On définit la distance ($d_c \in \mathbb{R}^+$) entre deux points du carré unitaire x et y dans $]0,1[^2$ comme la taille du plus petit cadran les contenant tous les deux, et zéro s'ils sont égaux :

$$d_c(x,y) = \begin{cases} 0 & \text{si } x = y \\ \inf_k(2^{-k} : \exists 0 \leq i,j < 2^k \wedge \{x,y\} \subset E^k_{i,j}) & \text{sinon} \end{cases}$$

Proposition 5.1.5. *d_c est une distance sur $]0,1[^2$. En d'autres termes, on a les trois propriétés suivantes :*

- *Symétrie :* $\forall x,y \in]0,1[^2, d_c(x,y) = d_c(y,x)$
- *Séparation :* $\forall x,y \in]0,1[^2, d_c(x,y) = 0 \Leftrightarrow x = y$
- *Inégalité triangulaire :* $\forall x,y,z \in]0,1[^2, d_c(x,y) \leq d_c(x,z) + d_c(z,y)$

Démonstration. Cette proposition est démontrée en trois étapes, une par propriété.

1. La symétrie découle immédiatement de la définition de d_c.
2. La séparation : $x = y \Rightarrow d_c(x,y)$ par définition. De plus, on note que d_c est supérieure ou égale la distance euclidienne divisée par $\sqrt{2}$ (notée d_e). De ce fait, $d_c(x,y) = 0 \Rightarrow d_e(x,y) = 0 \Rightarrow x = y$.
3. L'inégalité triangulaire se montre par un raisonnement par l'absurde pour les cas où x et y sont différents[1]. On pose, pour que cette inégalité soit fausse, que $d_c(x,z) < d_c(x,y)$ et que $d_c(z,y) < d_c(x,y)$ avec $d_c(x,y) = 2^{-k}$. Soit $E = E^k_{i,j}$, le plus petit cadran contenant x et y. D'après les deux inégalité précédentes, on note que x et z sont dans un cadran plus petit que E. Ils appartiennent donc à un cadran de la forme $E^{k+1}_{i',j'} \subset E$. De même pour y et z, on note qu'ils appartiennent à un cadran de la forme $E^{k+1}_{i'',j''} \subset E$. Or, un point ne peut pas appartenir à deux cadrans différents d'un même ordre (par définition). Puisque z appartient à ces deux cadrans, ils sont identiques. On a alors que $\{x,y,z\} \subset E^{k+1}_{i',j'}$ et donc $d_c(x,y) \leq 2^{-(k+1)}$. Il y a donc contradiction, l'inégalité triangulaire est donc vérifiée.

□

Définition 5.1.6. On définit la distance de Hausdorff entre deux parties de $]0,1[^2$ en utilisant la définition classique de la distance de Hausdorff mais en utilisant d_c pour calculer les distances entre points. La distance de Hausdorff entre ensemble est définie formellement de la manière suivante :

$$\forall A,B \subset]0,1[^2, d_H(A,B) = \max_{a \in A}(\min_{b \in B} d_c(a,b))$$

5.2 Utilisation du jeu du chaos pour définir une distance entre langages

Définition 5.2.1. On définit une distance entre langages en utilisant la distance de Hausdorff entre leur représentation par jeu du chaos respectives. Formellement, cette distance d_L est définie comme suit :

$$\forall L_1, L_2 \subset \Sigma^*, d_l(L_1,L_2) = d_H(CGR(L_1), CGR(L_2))$$
$$\forall L_1, L_2 \subset \Sigma^*, d_L(L_1,L_2) = max(d_l(L_1,L_2), d_l(L_2,L_1))$$

Proposition 5.2.2. *On interprète la valeur de $d_l(L_1,L_2)$ comme une estimation de la taille des mots présents dans les deux langages. Plus d_l est petit, plus la longueur des mots de L_2 contenu dans L_1 est grande. Formellement, on interprète d_c de la manière suivante :*

$$\forall L_1, L_2 \subset \Sigma^*, d_l(L1,L2) = 1/2^n \Leftrightarrow \forall w_2 \in L_2 \begin{cases} w_2 \in L_1 & si\ |w| < n_2 \\ \exists w \in \Sigma^n, w_1 \in L_1, w'_1, w'_2 \in \Sigma^* : w_1 = w'_1.w \wedge w_2 = w'_2.w & sinon \end{cases}$$

Démonstration.

[1] si $x = y$, elle est trivialement vraie puisque d_c est à valeurs positives.

1. Soit $|w_2| < n$ Quel que soit le point $y \neq y_w$, $d_c(y, y_w) > 1/2^n$. On a donc que $w \in L_1$.
2. Soit $|w_2| \geq n$
   Par définition de la distance d_c, on a l'équivalence avec la propriété suivante :

$$\exists w_1 \in L_1 \text{ et un cadran } E^n_{i,j} \text{ tels que } \{w_1, w_2\} \subset E^n_{i,j}$$

   Par définition des cadrans, on a l'équivalence avec la propriété suivante :

$$\exists w \in \Sigma^n, w'_1 \text{ et } w'_2 \in \Sigma^* : w_1 = w'_1.w \text{ et } w_2 = w'_2.w$$

□

5.3 Exemple de motif associés à des séquences d'ADN

La représentation par jeu du chaos de séquences de lettres ne permet pas d'interprétation intuitive de la distance entre une séquence et un langage. Le problème vient du fait qu'on ne considère que les préfixes de la séquence. Par exemple, la distance entre la représentation de la séquence $acgt$ et celle de Σ^* est de $1/2$ car les mots c, g et t ne sont pas préfixes de $acgt$.

On définit alors une variante de la représentation par jeu du chaos des séquence en considérant tous les sous-mots de la séquence. Pour une séquence S, on définit sa représentation par $CGR'(S) = CGR(suf(pre(S)))$. En figure 5.3.1, nous avons représenté les 500 000 premières bases de deux génomes. On remarque que ces représentations sont similaires à leur version classique. La seule différence étant dans le fait qu'il y a plus de points affichés, soulignant ainsi les motifs déjà repérés.

Cette variante de représentation confère à la distance entre séquence et langage plus de sens. La distance rend mieux compte de l'apparition des mots du langage au sein de la séquence (et non plus uniquement comme préfixes). La distance entre $acgt$ et Σ^* est alors de $1/4$ puisque tous les mots de taille 1 sont présent dans $acgt$.

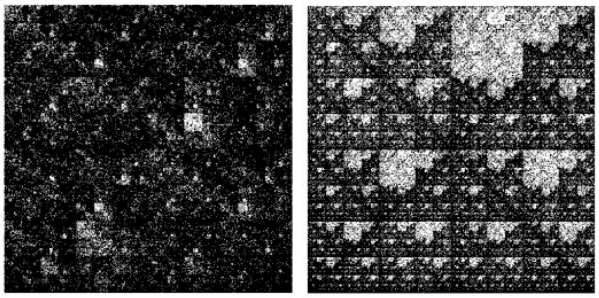

FIG. 5.3.1 – Représentation par variante du jeu du chaos des 500000 premières bases de deux génomes (E. Coli à gauche et l'Homme à droite).

Encore une fois, comme pour la distance de Hausdorff en général, la version non symétrique (d_l) donne plus d'informations que la distance elle-même (d_L). Par exemple, la distance d_l du le génome humain vers le langage modélisant la diagonale (ce langage contient tous les mots n'utilisant que a et g) est inférieur à $1/2^{10}$ (car dans nos représentations, tous les pixels de la diagonale sont noirs). On sait alors que tous les mots de $\{a, g\}^*$ de taille inférieur à 10 sont présents dans la séquence. Par contre, la distance du langage vers la séquence est de 1 car la séquence utilise les deux autres bases. De même, la distance entre la séquence et le langage interdisant cg est inférieure à $1/2^{10}$ alors que la distance dans l'autre sens est de $1/4$ (puisque la séquence, contient quand même une occurrence de cg.

Grâce à cette distance, on caractérise la présence d'un motif autosimilaire en terme d'inclusion de langage régulier dans la séquence. En effet, étant donné un langage régulier, on détermine automatiquement sa représentation. Le fait que les motifs de la représentation du langage soit marqués dans la représentation d'une séquence se traduit par l'inclusion de tous les mots de taille au moins 10 du langage régulier dans la séquence. Ainsi, si un motif est marqué dans une séquence, cela indique que les mots du langage représenté par le motif sont tous présent jusqu'à une certaine taille dans la séquence.

Bien que cette représentation soit plus intuitive pour l'interprétation de la distance, on se bornera à tracer les représentations classiques. Tout d'abord, la variante de représentation est beaucoup plus coûteuse en calcul (pour un mot de taille n, on doit tracer $n(n+1)$ points au lieu de n dans la version classique). Ensuite, dû au contraintes matérielles, on ne peut pas différencier les images de deux mots ayant un suffixe commun de plus de 10 lettres. Les motifs visualisés dans les deux représentations seront donc les mêmes. Enfin, les représentations concernant des séquences d'ADN, et donc des mots finis contenant du bruit et des imperfections, l'erreur due à l'utilisation de la représentation classique par rapport à la variante est faible.

Chapitre 6

Visualisation et filtrage de motifs statistiques

D'un point de vue statistique, la représentation par jeu du chaos d'une séquence permet de voir la distribution des fréquences d'apparition des sous-mots de la séquence. En effet, on sait que les cadrans regroupent tous les mots de même suffixe ($\forall w_1, w_2 \in \Sigma^*, y_{w_1 w_2} \in U_{w_2}$). Ainsi, dans un cadran (U_w par exemple), il y a autant de points que d'occurrences de w dans la séquence. Un cadran fortement rempli représente un mot très représenté dans la séquence et, à l'inverse, un trou (un cadran vide) représente un mot très peu représenté. Il est donc possible de se rendre compte visuellement de la distribution des fréquences d'apparition des mots dans la séquence.

6.1 Représentation par cadrans, variante du jeu du chaos

C'est ainsi qu'on définit une autre variante de représentation par jeu du chaos, la représentation par cadrans. Cette représentation consiste à dessiner les cadrans des mots d'une longueur choisie et de les colorer en fonction de la fréquence d'apparition du mot dans la séquence (par exemple en nuances de gris, telles que plus un mot est représenté, plus la couleur du cadran est blanche). Dans la figure 6.1.1, nous avons regroupé huit représentations par cadrans du génome humain, pour des longueurs de mots de 1 (en haut à gauche) à 8 (en bas à droite). Ces représentations permettent de connaître les fréquences d'apparition des mots de taille fixe relativement aux autres mots de même taille. Ainsi, sur la première figure (en haut à gauche), on représente les quatre cadrans principaux et on remarque que les mots "c" et "g" sont moins fréquents que "a" et "t". Dans la deuxième (à sa droite), on note le trou du mot "cg" qui est bien moins fréquent que le reste des mots de longueur 2. Enfin, pour les mots de longueur 8, on obtient une figure similaire aux autres représentations de cette séquence.

Définition 6.1.1. La représentation par cadrans d'ordre k d'une séquence $S \in \Sigma^*$ consiste à afficher les cadrans U_w ($w \in \Sigma^k$) dont la couleur est fonction de la fréquence d'apparition de w dans S noté $\tilde{p}_S(w)$ et est donnée par la formule suivante :

$$\forall S, w \in \Sigma^*, \tilde{p}_S(w) = |U_w \cap CGR(S)|/(|S| - k + 1)$$

Cette version de représentation permet mieux de rendre compte des motifs comme étant des ensembles de mots exceptionnels. Les mots exceptionnels sont des mots dont la fréquence observée est différente de la fréquence attendue (qu'on détermine par un modèle). Dans ce cas, un filtrage par une représentation par cadrans correspondant à un modèle permet de mettre en évidence les mots exceptionels. De plus, dans la version classique du jeux du chaos, dû aux contraintes matérielles, on ne peut pas faire la différence entre les images de mots de suffixe commun plus long que 10 lettres (car ces images sont sur le même pixel de l'écran). Si tous les mots de taille 10 sont présent dans le langage, on ne peut plus savoir, avec le jeu du chaos, si une partie de ceux-ci sont sur-représentés. Ce cas de figure n'est pas gênant dans la représentation par cadrans puisqu'on n'affiche pas les images des mots, mais leur nombre dans les cadrans. C'est pour cette raison que les diagonales de la représentation du génome humain sont plus marquées dans la représentation par cadrans [Fig. 6.1.1] que dans la représentation classique [Fig. 3.3.1].

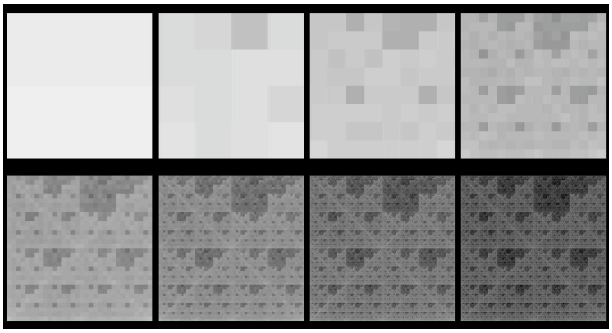

FIG. 6.1.1 – Représentation par cadrans (pour des ordres de 1 à 8) du chromosome 14 de l'Homme. Plus les cadrans sont noirs, moins le mot est représenté, plus il est clair, plus le mot est présent dans le chromosome.

6.2 Caractérisation des motifs statistiques par automates probabilistes

De même que nous avions utilisé des automates complets déterministes pour caractériser les structures fractales de la représentation classique, nous utilisons des automates probabilistes pour caractériser les motifs de la représentation par cadrans. Les automates classiques sont des outils pratiques et bien maitrisés pour définir des langages, mais ils ne permettent pas de rendre compte de fréquences d'apparitions. Pour un automate classique, un mot apparait ou pas, mais on ne sait pas combien de fois. Pour remédier à ce manque d'informations, on utilise des automates probabilistes. Ces automates sont similaires aux premiers (dans leur structure), mais permettent de nuancer la présence (et l'absence) d'un mot en définissant une probabilité d'apparition des mots.

Il n'existe pas de définition rigoureuse commune des automates probabilistes. Les automates probabilistes sont en fait un moyen plus visuel de représenter les chaînes de Markov [13, 14, 15, **?**] et la définition de l'automate probabiliste varie en fonction de l'utilité qu'on en a et de la façon dont on définit les chaînes de Markov. Nous avons choisi la représentation des chaînes de Markov issues de [15] pour construire nos automates.

Définition 6.2.1. Une chaîne de Markov d'ordre k sur un alphabet Σ est une suite de symboles $(X_i)_{i\geq 0}$ telle que la probabilité d'apparition d'un symbole à une étape n ne dépend que de la valeur des k derniers symboles apparus $(p(X_n = x|X_0X_1...X_{n-1}) = p(X_n = x|X_{n-(k+1)}...X_{n-1}))$.

Définition 6.2.2. On définit un automate probabiliste A comme un quintuplet (E, Σ, I, T, P) dont les éléments ont la signification suivante :

- E est l'ensemble (fini) des états de l'automate.
- Σ est l'alphabet de l'automate (ensemble fini de lettres).
- $I : E \to [0, 1]$ est la probabilité qu'un état soit initial.
- $T \subset E \times \Sigma \times E$ est l'ensemble des transitions. (e_i, α, e_f) signifie qu'il y a une transition partant de e_i, arrivant sur e_f et étiquetée par α. Ces transitions doivent respecter la condition suivante : $\forall e_i, |\{(e_i, \alpha, e_f, p) \in T\}| = 1$. Ainsi, pour chaque état, en lisant une lettre, le choix de l'état cible est toujours déterminé.
- $P : E \times \Sigma \to [0, 1]$ est la probabilité de transition $p(e, x)$ qui donne la probabilité de la transition commençant sur e et étiquetée par x.

On définit alors une fonction de transition $\delta(e, x)$, qui étant donné un état e et une lettre x, donne l'état dans lequel on aboutit par la transition étiquetée par x et commençant sur e. On a ainsi $\delta(e, x) = e_2$ si et seulement s'il existe $p \in [0, 1]$ tel que $(e, x, e_2, p) \in T$. Par facilité, on note $\delta(e, w) : w \in \Sigma^*, e \in E$, l'état sur lequel on aboutit en partant de e et en lisant le mot w.

Les automates probabilistes permettent ainsi d'attribuer une probabilité à chaque mot de Σ^k. La probabilité d'un mot est, pour chaque état de départ e, le produit des probabilités des transitions correspondantes aux lettres du mots (en commençant le chemin sur e). Ainsi, pour tout mot w, on caclule

la probabilité de w, $p(w)$ par la formule suivante :

$$p(w) = \sum_{e \in E} \left(I(e) \prod_{i=1}^{|w|} p(\delta(e, w_{1..i-1}), w_i) \right)$$

où $w_{1..i-1}$ est le préfixe de w de taille $i-1$.

Proposition 6.2.3. *Toutes chaînes de Markov d'ordre d est exprimable par un automate probabiliste (appelé automate markovien d'ordre k).*

Démonstration. On va montrer ici comment construire de tels automates à partir d'une chaîne de Markov d'ordre d donnée.

1. L'alphabet de l'automate est le même que pour la chaîne de Markov.
2. On construit un état de l'automate pour chaque suite de d symboles de la chaîne de Markov. Il y a donc $|\Sigma|^d$ états dans l'automate et pour chaque mot w, on note e_w, l'état qui le représente.
3. La probabilité d'un état initial est égal à la probabilité d'apparition de la suite de symboles correspondants d'après le modèle de Markov.
4. Pour chaque mot w de longueur d, avec w', le suffixe de longueur $d-1$ de w, on définit les transitions de la manière suivante : $(e_w, x, e_{w'x})\ \forall x \in \Sigma$ où $p(e_w, x) = p(x|w_0...w_d)$ est la probabilité que x apparaisse sachant que les d derniers symboles (w) sont sortis d'après la chaîne de Markov.

□

Proposition 6.2.4. *Tout automate probabiliste n'est pas modélisable par une chaîne de Markov d'ordre k fixé.*

Démonstration. Nous allons ici montrer un exemple d'automate qui ne peut pas être modélisé par chaîne de Markov.

Soit A l'automate contenant les éléments suivants :

1. Il y a trois états : $\{e_1, e_2, e_3\}$,
2. L'alphabet contient deux symboles : $\{0, 1\}$,
3. Seul l'état e_1 est initial ($I(e_1) = 1$),
4. Les six transitions sont les suivantes :

$(e_1, 0, e_2)$ de probabilité 0.9	$(e_1, 1, e_3)$ de probabilité 0.1
$(e_2, 0, e_1)$ de probabilité 0.1	$(e_2, 1, e_3)$ de probabilité 0.9
$(e_3, 0, e_1)$ de probabilité 0.5	$(e_3, 1, e_3)$ de probabilité 0.5

Cet automate modélise un langage où les mots contenant des suites paires de 0 sont plus probables que ceux contenant des suites impaires de 0. La probabilité d'apparition d'un 0 dépend d'un nombre arbitrairement grand de 0 déjà lus.

De ce fait, on ne peut modéliser cette contrainte par chaine de Markov d'ordre k fixé. □

La représentation par cadrans permet donc, en plus des séquences, de représenter les automates probabilistes (plus généraux que les chaînes de Markov). Étant donné un ordre k, pour chaque $w \in \Sigma^k$, on affiche U_w d'après la probabilité d'apparition de w calculée d'après l'automate.

6.2.1 Exemple de représentation d'automates probabilistes

Pour permettre de mieux appréhender cette représentation des automates probabilistes, en lien avec celle des séquences, nous détaillons ici deux exemples. Tout d'abord, nous nous penchons sur l'automate modélisant les trous du génome humain dûs à la contre sélection du mot *cg*. Enfin, nous discuterons de l'automate probabiliste modélisant la présence des diagonales (également présentes dans la représentation du génome humain).

Trous en cascade : Comme nous l'avons déjà vu, la présence des trous dans la représentation du génome humain est due à une faible fréquence d'apparition du mot cg dans ce génome. On peut modéliser cette caractéristique à l'aide de l'automate probabiliste donné en figure 6.2.1. Cet automate est définis par deux états (1 et 2) dont la probabilité d'être initial est de $0,5$. L'état 2 regroupe les mots finissant par c, l'état 1 regroupe les autres mots. Les transitions sortant de l'état 1 sont équiprobables. La probabilité des transitions sortant de 2 en lisant un a, c ou un t sont de $0,33$ et celle de la transition en lisant un g est de $0,01$. Ainsi, cet automate défavorise les mots contenant gc.

La représentation de cet automate est donnée en figure 6.2.3. On remarque la structure hiérarchique de trous déjà observée dans la représentation du génome humain. On note également que, contrairement à la représentation par IFS du langage interdisant cg [Fig. 4.2.3], le trous contient aussi la même structure de trous. Cette zone peu représentée de l'image répond à la même structure que le reste de l'image. Cette observation est valable aussi pour la représentation du génome humain par cadrans [Fig. 6.1.1] où on distingue aussi cette structure dans les trous.

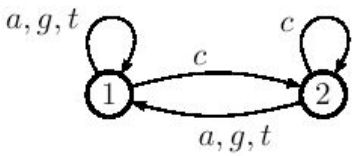

FIG. 6.2.1 – Automate modélisant une particularité du mot cg par rapport aux autres mots. Les probabilités des transitions partant de l'état 1 sont de $0,25$, celle partant de l'état deux en lisant a, c ou t sont de $0,33$ et en lisant un g est de $0,01$. La probabilité d'être initial est équiprobable pour tous les états.

Diagonales : Les diagonales observées dans la représentation du génome humain sont également caractérisables par un automate probabiliste. Une diagonale seule est l'image du langage où soit seul a et g soit seul c et t (suivant la diagonale) sont autorisés. Une image où les diagonales sont marquées représente une séquence contenant beaucoup de ces mots privilégiants deux lettres sur les quatre. L'automate probabiliste correspondant [Fig. 6.2.2] contient deux états. L'un pour les mots finissant par a ou g, l'autre pour les mots finissant par c ou t. La probabilité des transitions bouclant sur un état[1] est de $0,4$, et celle des transitions changeant d'états est de $0,1$.

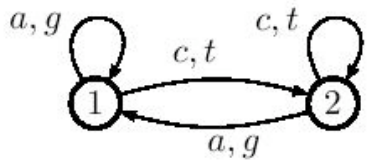

FIG. 6.2.2 – Automate modélisant une particularité des mots contenant des suites de a ou g et des suites de c ou t. Les probabilités des transitions qui bouclent sont de $0,4$ et celles des transitions qui changent d'états sont de $0,1$.

La représentation de cet automate est donnée en figure 6.2.3. On note les deux diagonales principales, déjà observée dans le génome humain. On note également que cette structure du carré contenant deux diagonales marquées est reproduite sur tous les cadrans. Ce phénomène est présent aussi dans la représentation du génome humain, mais plus discrète de par la présence des trous qui occultent le reste. Encore une fois, on n'aurait pas pu modéliser cette structure auto-similaire et très ramifiée avec un automate classique et son L-système car dans cette figure, tous les mots sont autorisés (le langage correspondant est Σ^*).

[1] Une transition boucle sur un état, si cet état est source et cible de la transition.

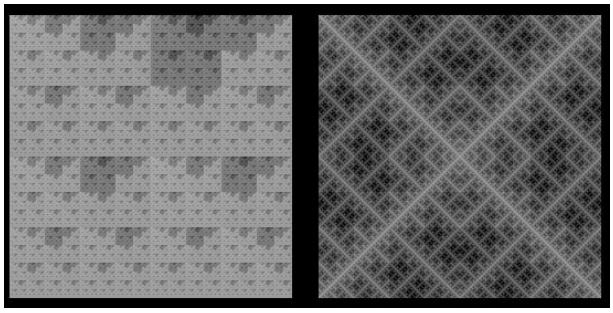

FIG. 6.2.3 – Représentation d'automates probabilistes. Le langage défavorisant "gc" est à gauche, le langage favorisant les suites de a+g ou de c+t est à droite.

6.3 Suppression des motifs dans les représentations par filtrage

Le défaut de ces représentations par jeu du chaos, c'est que les gros motifs fractals cachent tous les autres et il serait intéressant de pouvoir les supprimer pour voir ce qu'il y a derrière. En fait, ces motifs, présents dans toute la figure, occultent les autres motifs plus discrets. Dans la représentation du génome humain, il est difficile de distinguer autre chose que les diagonales et les trous en cascades qui parsèment la figure. Or, comme nous le verrons plus loin, il n'y a pas que ces deux motifs dans la figure.

L'idée, pour supprimer des motifs choisis, est de filtrer la représentation par le modèle statistique correspondant au motif. Cette idée est soulevée dans [16], où les auteurs filtrent une représentation similaire au jeu du chaos, par un modèle de Markov d'ordre 1. Nous définissons ici cette notion de filtrage et l'étendons en utilisant les automates probabilistes.

Une représentation d'une séquence filtrée par un modèle consiste à afficher les cadrans dans une couleur en fonction du rapport entre la fréquence d'apparition d'un mot d'après le modèle ($\tilde{p}(w)$) et la probabilité d'apparition du mot ($p(w)$). Les mots exceptionnels ont un rapport éloigné de 1 et suivant ce rapport, on peut déterminer si le mot est sur-représenté ($\tilde{p}(w)/p(w) > 1$) ou sous-représenté ($\tilde{p}(w)/p(w) < 1$). Nous avons choisi de représenter en blanc les cadrans dont le rapport est proche de 1, en nuances de bleu ceux dont les cadrans ont leur rapports supérieurs à 1 et en rouge les autres. Plus la couleur est vive, plus le mot est exceptionnel.

Cependant, avant de filtrer une représentation d'une séquence par un modèle, il est nécessaire d'estimer les paramètres du modèle pour qu'ils *collent* à la séquence. Dans notre cas, cette estimation se fait sur la loi d'état initial et sur les probabilités de transitions. La méthode utilisée consiste à parcourir l'automate en lisant la séquence (S). Pour chaque état, on commence une lecture de la séquence pour voyager dans l'automate en parcourant les transitions correspondant aux lettres de la séquence. On dispose alors des deux compteurs suivants : $N(e)$ qui donne le nombre de fois ou l'on est passés par e, et $N(e, x)$ qui donne le nombre de fois où l'on a lu un x en étant dans e. La probabilité qu'un état soit initial est donnée par $p(e) = N(e)/|S|$. De même, la probabilité d'une transition débutant sur e étiquetée par x est donnée par $N(e, x)/N(e)$.

Cette technique est utilisée pour filtrer la représentation d'une séquence par des modèles de plus en plus sophistiqués. On commence par filtrer la représentation par un modèle de probabilités uniformes. Ce modèle permet de voir les premiers motifs. On identifie alors un motif et son automate probabiliste dont on estime les paramètres. On filtre alors la représentation de la séquence par ce nouveau modèle, dévoilant d'autres motifs. On identifie alors un de ces motifs et son modèle et on filtre la représentation par l'union de ces modèles. En itérant ce processus, on obtient une suite de modèles dont l'union converge vers un modèle plus proche de la séquence.

Dans la figure 6.3.1, nous donnons une suite de filtrages de la représentation du génome humain par modèles de plus en plus précis (de gauche à droite). Nous avons, dans un premier temps, filtré la représentation par un modèle de loi uniforme. Ensuite, nous avons choisi un modèle de loi non-uniforme mais sans dépendances entre lettres successives. Il s'agit d'une chaîne de Markov d'ordre 0. Pour continuer, nous avons supprimé les diagonales, faisant ressortir les trous. Enfin, nous avons supprimé les trous. On note qu'il reste encore des motifs fractals (des trous qui se reproduisent un peu partout dans la figure) et on pourrait continuer le processus.

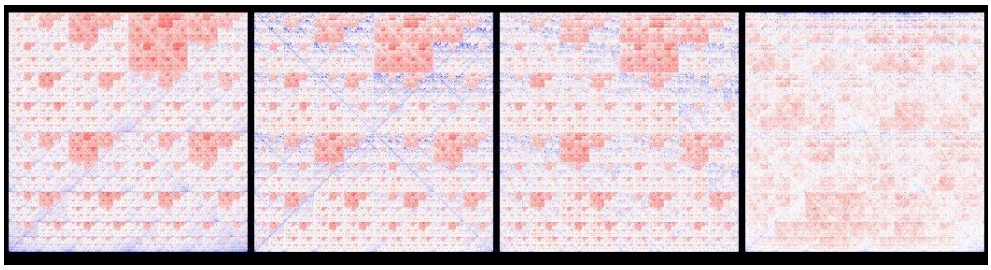

FIG. 6.3.1 – Filtrages successifs de la représentation du génome humain par des modèles de plus en plus précis. De gauche à droite, les modèles ajoutés sont : probabilités uniformes, un automate à un seul état (équivalent à une chaine de markov d'ordre 0), le modèle des diagonales et enfin le modèle des trous en cascade.

Chapitre 7

Utilisation des motifs dans la discrimination des génomes

Dans [12, 17], il a été suggéré d'utiliser le jeu du chaos pour discriminer des séquences d'après leur représentation par cadrans. A partir de la représentation par cadrans d'ordre 8, l'idée est de caractériser la séquence représentée par un vecteur de $\mathbb{R}^{4^8}$ contenant les fréquences d'apparition des mots de Σ^8. Ils définissent ensuite la distance entre séquences comme la distance euclidienne entre les vecteurs qui les caractérisent.

Au lieu d'utiliser des vecteurs de $\mathbb{R}^{4^k}$, nous utilisons des automates probabilistes markoviens. La représentation par cadrans d'ordre k est une manière de visualiser une distribution de probabilités sur Σ^k. De même, une chaîne de markov d'ordre $k-1$ permet de caractériser cette même distribution sans perdre d'information. On utilise ainsi l'automate markovien correspondant à la chaîne de markov pour caractériser cette distribution et donc, la séquence.

Étant donné un automate probabiliste, on définit alors la distance entre deux séquences comme étant la distance entre les paramètres des automates adaptés à ces séquences. Cette distance n'est pas fixée rigoureusement et on peut très bien utiliser différentes définitions de distances suivant les besoins. Par exemple, pour des automates à n états, on définit des vecteurs correspondants dans $\mathbb{R}^{4n}$ regroupant les probabilités des transitions. On peut alors définir la distance entre automates comme la distance euclidienne entre leur vecteurs respectifs. Cette distance a l'avantage de correspondre à la distance données dans [12].

Pour éviter de gérer trop de variables, nous définissons une opération de fusion sur les états d'automates probabilites. Le problème des automates markoviens d'ordre 7, c'est le grand nombre de paramètres (ils contiennent 4^7 états et donc 4^8 transitions). Or on a vu que les structures observées dans les représentations ne dépendaient que d'un nombre faible de paramètres (les trous de la représentation du génome humain sont dus à un automate à seulement deux états et huit transitions). L'idée de la réduction est de se dire que certains états de l'automate contiennent la même information et qu'il n'est pas utile de les différentier. La réduction d'automate est l'action de fusionner ces états équivalents pour en diminuer le nombre.

Définition 7.0.1. On définit la fusion d'un ensemble d'états $\{e_1, ..., e_k\}$ dans un automate $A = (E, \Sigma, I, \delta)$ comme l'action de remplacer ces états dans A par un nouvel état e_0 (tout en modifiant les autres éléments pour garder une certaine cohérence).

Pour unifier un ensemble d'états, ceux-ci doivent respecter la condition suivante :

$$\forall x \in \Sigma, [\forall i \in \{1..k\}, \delta(e_i, x) \in \{e_1, ..., e_k\}] \text{ ou } [\exists e \in E : \forall i \in \{1..k\}, \delta(e_i, x) = e]$$

Le nouvel automate A' sera défini par les éléments suivants :

1. les états : $E' = e_0 \cup E \backslash \{e_0, ..., e_k\}$,
2. l'alphabet : il reste le même,
3. la probabilité d'être initial : elle est inchangée pour les états de $E' \cap E$. Pour e, elle vaut $I'(e) = \sum_{i=0}^{k} I(e_i)$,

4. Les transitions n'utilisant aucun état à fusionner sont conservées dans A'. On modifie la destination des transitions de A débutant sur un état de $E' \cap E$ et finissant sur un des états de $\{e_0, ..., e_k\}$ en la remplaçant par e_0. Enfin, on construit les quatre transitions partant de e_0 de la manière suivante :

$$\forall x \in \Sigma, (e_0, x, e_f, \sum_{i=0}^{k} p(e_i, x).p(e_i)) \in \delta'$$

Avec $e_f = e_0$ si les transitions de A partant des états à fusionner aboutissent sur d'autres états à fusionner, et $e_f = \delta(e_1, x)$ sinon. La condition nécessaire à la fusion garantit que ces transitions soient construites sans ambiguïté.

Fusionner ainsi des états permet de mettre l'accent sur des motifs plutôt que sur la figure entière. Par rapport à un automate markovien d'ordre 7, qui prend en compte la totalité de la représentation par cadrans d'ordre 8, un automate dont une bonne partie des états sont fusionnés permet de se focaliser sur une particularité de l'image. Par exemple, on note que les automates markoviens d'ordre k sont obtenus par fusions d'états à partir d'automates markoviens d'ordre $k' > k$. De même, l'automate représentant les diagonales est obtenus par fusions d'états de l'automate markovien d'ordre 1 (les états de l'automate markovien correspondants aux mots a et g sont fusionnés ainsi que les états des mots c et t). Ainsi, cet automate est obtenu par fusions d'états de l'automate markovien d'ordre 8 (ou plus).

La fusion d'états est une technique similaire à l'analyse en composantes principales. Dans [12], les auteurs ont recourus à une analyse en composantes principales pour permettre de sélectionner un sous-ensemble de dimensions pour leurs vecteurs de $\mathbb{R}^{4^8}$. Cette analyse leur a montré que la somme des fréquences du mot c plus celle du mot g est suffisante pour discriminer efficacement un grand nombre de génomes. De même, dans [18], les auteurs utilisent la distribution de fréquences d'apparition des mots de 3 lettres et, par une analyse similaire, réduisent leur nombre de dimention. Dans cet article aussi, ils retiennent la fréquence de c plus la fréquence de g.

Cette restriction à un nombre plus restreint de variables dans leur modèle s'assimile, pour nos automates, à réduire le nombre d'états et de transitions. En effet, un automate à un seul état permet de modéliser toute particularité des fréquences d'apparitions des lettres et est obtenu en fusionnant tous les états des automates markoviens.

Chapitre 8

Perspectives

8.1 Localité

Un inconvénient de la représentation par jeu du chaos est qu'elle ne permet pas de prise en compte de localité au sein de la séquence. La figure tracée par le jeu du chaos pour une séquence donnée permet de voir des motifs présents tout le long de la séquence. Par contre, on ne peut pas savoir si ces motifs sont plus ou moins présents à divers endroits de la séquence. Par exemple, dans la représentation du génome humain, il y a des points dans le cadrans de cg , or on ne peut pas savoir si ces points sont des images de mots proches dans la séquence ou dispersés uniformément tout au long de celle-ci.

Pour permettre de prendre en compte la localité dans les séquences tout en utilisant le jeu du chaos, nous avons eu l'idée d'effectuer des représentations sur des fenêtres glissantes le long de la séquence. Le principe consiste à découper la séquence en morceaux de taille fixe. Ensuite, on effectue une représentation par jeu du chaos de chaque morceau séparément. On dispose alors d'une dimension supplémentaire caractérisant la localité.

Pour afficher ces données en deux dimensions, nous avons projeté le jeu du chaos sur une dimension. Pour cela, les points de contrôles ont les valeurs suivantes : $p_a = 0$,$p_c = 1$,$p_g = 2$ et $p_t = 3$. Les transformations sont alors de la forme : $u_x(y) = y/4 + p_x$. Le point initial est $e_\epsilon = 2$. De cette manière, le jeu du chaos d'une séquence produit une distribution de points le long du segment $[0, 4]$. Pour permettre plus de lisibilité, nous avons choisi, sur le segment $[0, 4]$ de ne pas afficher les points tracés par le jeu du chaos, mais d'afficher la densité de points par niveaux de gris sur des petits bouts de celui-ci. On peut alors juxtaposer les représentations des morceaux du génome le long de la deuxième dimension. La figure obtenue permet de se rendre compte de la fréquence d'apparition de chaque mot donné (en ordonnée) à différents endroits dans la séquence (en abscisse) [Fig. 8.1.1].

Dans ces représentations, on peut alors voir les changements de distributions de fréquences d'apparition des mots. En effet, si une zone de la séquence n'obéit pas au même modèle probabiliste que le reste de la séquence, avec une longueur de fenêtre glissante adaptée (de l'ordre de la longueur de la zone), on remarquera un changement de motif dans la représentation à l'endroit de la zone dans la séquence. C'est le cas dans le génome de la bactérie Sulfolobus acidocaldarius dont la représentation est donnée figure 8.1.1. Dans cette représentation, on note, à droite de l'image, au moins quatre zones de distributions de fréquences différentes. Ils s'agit des colones noires comportant quelques traits blancs horizontaux. Ces bandes noires et leur lignes blanches correspondent à des zones du génome de la bactérie contenant un mot (généralement de longeur 20) très répété, séparé par des longs mots uniques dans la séquence.

Cette représentation permet de visualiser aisément la présence de crispr [19]. Les crispr's sont justement ces zones dans les génomes constituées d'un mot très répété séparant des mots uniques. Ces zones sont très reconnaissables dans les représentations par jeu du chaos sur des fenêtres glissantes. En effet, ces zones, de par l'aspect très répété d'un mot contiennent quelques points très brillants, et le reste de la colonne de la zone est noir puisqu'en dehors de ces mots, le reste est quasiment aléatoire.

La recherche de crispr étant un thème de recherche de l'équipe Symbiose de l'IRISA [20], cette méthode de visualisation des séquences d'ADN fournit un moyen visuel complémentaire de présenter leur résultats. En effet, on dispose actuellement de moyen efficace de détecter les Crispr, mais leur visualisation n'est pas toujours aisée. Cette représentation par jeu du chaos sur des fenêtre glissante, bien qu'elle ne permet

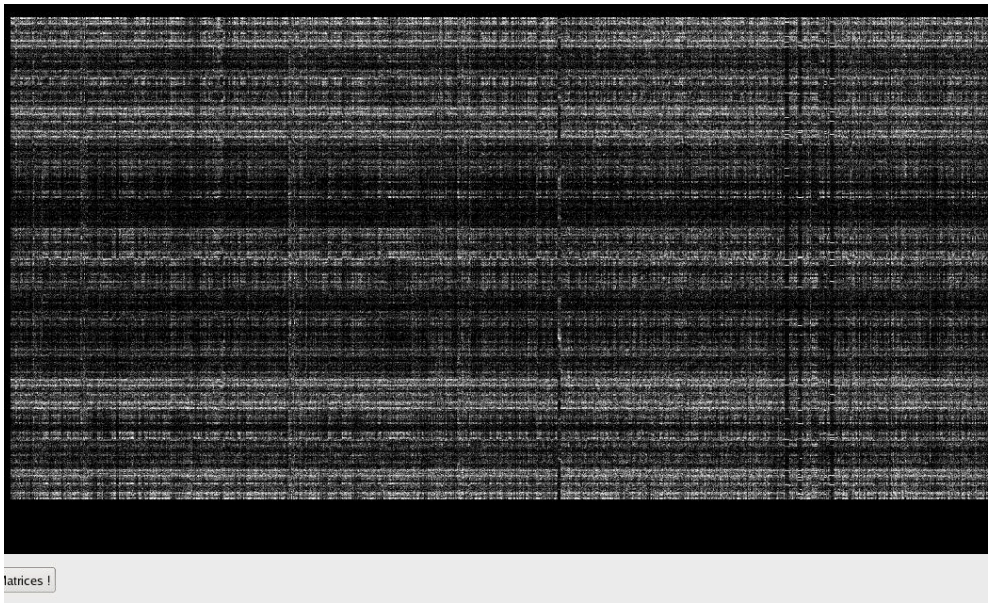

FIG. 8.1.1 – Représentation par jeu du chaos sur des fenêtres glissantes du génome de la bactérie Sulfolobus acidocaldarius. Les fenêtres font 1500 bases de long.

pas efficacement de découvrir les crispr, est une méthode complémentaire permettant de les visualiser.

8.2 Symétries statistiques

La réduction d'automates nous a permis de résumer les caractéristiques statistiques les plus visibles d'un langage. Ces caractéristiques ont un pouvoir discriminant et servent à distinguer les genomes d'organismes différents ou les parties du même génome ayant des rôles différents. Dans cette section nous analysons des caractéristiques simples communes à des classes de génomes ou de langages. Ces caractéristiques sont les symétries. En général, la présence d'une symétrie renseigne sur les mécanismes générateurs du langage et pour cette raison nous les accordons une attention toute particulière. Notons que les symétries qui abondent en cristallographie en relation avec la structure de la matière, ne sont qu'insuffisamment étudiées en relation avec la structure de l'ADN. Parmi les symétries de l'ADN nous pouvons citer le très recent modèle du solénoide [21] pour la structure de la chromatine. Selon ce modèle, les génes co-régulés se répètent périodiquement sur la séquence.

Nous attirons l'attention ici sur une autre symétrie statistique surprenante de la structure primaire de l'ADN. Cette symétrie est également signalée dans un travail récent [22].

Commençons par un avertissement. Il faut en général distinguer entre les symétries de la représentation par cadrans et les symétries de l'ADN. Même si les symétries de la représentations, peuvent, lorsque cette représentation est bien choisie, renseigner sur les symétries de la séquence, il est vrai que la représentation peut contenir plus de symétries que la séquence. Un cas typique est la symétrie fractale statistique de la représentation par cadrans, qui est le résultat de la méthode de construction et en aucun cas ne renseigne sur la fractalité de l'ADN. En effet, la mesure de probabilité représentant la fréquence de mots correspondants aux cadrans peut être calculée récursivement pour un modèle de Markov d'ordre d. Étant donné un mot $w = w_1...w_k$, soit $p(w)$ la probabilité d'apparition de ce mot. Alors on a la formule suivante :

$$p(w) = \frac{p(w_1..w_{k-1}).p(w_2...w_k)}{p(w_2...w_{k-1})}, \forall k > d-1$$

En pratique on remplace la probabilité $p(w)$ par la fréquence $f(w)$ observée de w. La formule a comme conséquence la fractalité de la représentation par cadrans même pour des chaînes de Markov.

De plus, une fois que les fréquences de mots de petite longueur sont connues, on peut estimer les fréquences des mots plus longs par la formule. Dans le cas général (même non-Markovien), la formule donne l'estimateur de maximum de vraisemblance de $p(w)$.

$$\tilde{f}(w) = \frac{f(w_1..w_{k-1}).f(w_2...w_k)}{f(w_2...w_{k-1})}$$

Tout écart entre l'estimateur $\tilde{f}(w)$ et la valeur observée $f(w)$ pourrait renseigner sur l'importance du mot. Les mots tels que $f(w)/\tilde{f}(w) > 1$ sont importants. Les auteurs de [22] proposent d'utiliser ce rapport comme mesure du contenu informationel du mot.

Pour visualiser cette caractéristique pour les mots de taille fixe d'un génome, nous avons effectué des représentations par cadrans de ce rapport. Il s'agit en fait d'afficher une représentation par cadrans filtrée par un modèle, ce que nous avions déjà défini dans les sections précédantes. En figure 8.2.1, nous avons affiché la représentation par cadrans filtrée d'après le modèle de [22] pour le génome de E. Coli. On note que mis à part des points parsemés, l'image est très claire, indiquant que le rapport entre fréquences estimées et fréquences réelles est proche de 1 pour tous les cadrans.

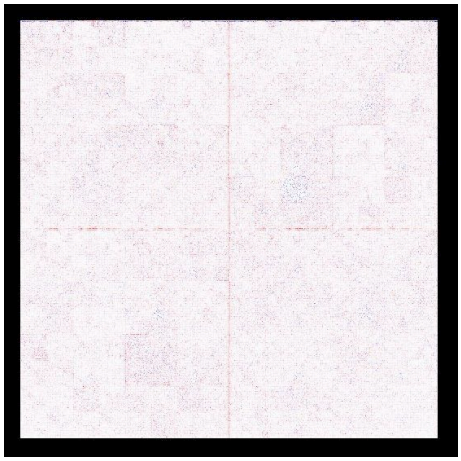

FIG. 8.2.1 – Représentation du rapport $f(w)/\tilde{f}(w)$ pour les mots de longeur 8 du génome de E. Coli.

Pour tout mot $w = w_1..w_k$ définissons son inverse complémentaire comme $PC(w) = w_k^c w_{k-1}^c ... w_1^c$, où w_k^c est la base complémentaire à w_k. Lorsque sur un brin de l'ADN on lit le mot w alors sur le brin complémentaire on lit le mot $PC(w)$ étant donné que les orientations des deux brins sont opposées. Il est donc tout a fait normal d'avoir autant de mots w que $PC(w)$ lorsqu'on analyse les deux brins du génome, mais rien ne permet d'affirmer que ceci est vrai aussi lorsqu'on analyse un brin seulement.

Dans le travail [22] la surprise vient de l'égalité approchée entre les rapports $f(w)/\tilde{f}(w)$ pour w et pour $PC(w)$, pour une analyse uni-brin, pour les mots de longueur inférieure à 8 qui ont un contenu informationnel fort. Ce résultat suggère une symétrie palindromique complémentaire statistique pour l'ensemble de mots courts sur un brin d'ADN.

Pour vérifier ces résultats, nous avons utilisé des représentations par jeu du chaos. Le jeu du chaos présente en effet une symétrie naturelle entre chaque mot et son complémentaire. En effet, pour un cadran U_w, le cadran U_{w^c}, représentant le complémentaire de w, est le symétrique de U_w par un axe de symétrie vertical découpant la figure en deux. Nous avons alors effectué deux représentations par cadrans filtrée pour une séquence. La première concerne la séquence elle-même, la deuxième concerner l'inverse de la séquence (la séquence lue en sens inverse). En disposant les images côte à côte, il y a un axe de symétrie vertical les séparant permettant d'envoyer tout cadran U_w d'une représentation vers le cadran représentant le cadran U_{w^c} de la séquence inverse (représentant donc $U_{CP(w)}$).

Dans la figure 8.2.2, nous avons dessiné ces deux représentations pour le génome de E. Coli. On remarque que cette figure est symétrique par un axe vertical séparant les deux représentations. Cette symétrie est la conséquence directe de la symétrie du rapport $f(w)/\tilde{f}(w)$ entre chaque mot w et son complément palinbdromique $CP(w)$.

Cependant, c'est symétrie ne concerne pas uniquement le rapport $f(w)/\tilde{f}(w)$, mais aussi la fréquence réelle d'apparition des mots de petite taille. En effet, en effectuant deux représentations par cadrans non filtrées, une pour la séquence et une pour son inverse, on observe toujours cette symétrie verticale entre

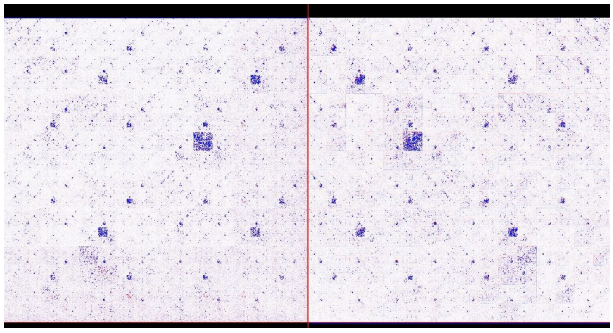

FIG. 8.2.2 – Représentation des rapports $f(w)/\tilde{f}(w)$ des mots du génome de E. Coli pour la séquence (à gauche) et son inverse (à droite).

les deux représentations. En figure 8.2.3, nous avons dessiné ces deux représentations pour le génome de E. Coli.

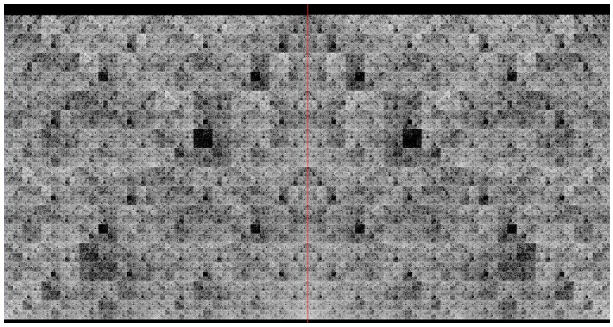

FIG. 8.2.3 – Représentations par cadrans d'ordre 8 de la séquence de E. Coli (à gauche), et de son inverse (à droite).

Il s'agit là d'un résultat nouveau et plus général que celui sur le rapport $f(w)/\tilde{f}(w)$. En effet, à notre connaissance, la littérature ne fait pas mention à cette symétrie statistique entre la fréquence d'apparition d'un mot et de son complément palindromique.

Conclusion

Le jeu du chaos permet de représenter des génomes par des images généralement fractales. Cependant, on ne disposait pas de réelles justifications ni d'explications au sujet de ces structures fractales. On ne savait pas si ces structures étaient dues à la méthode de représentation ou si la séquence contenait une certaine fractalité.

Nous savons maintenant que la présence de structures fractales dans la représentation d'une séquence d'ADN ne traduit pas de propriété fractale de la séquence, mais le fait qu'elle contient un certain nombre de langages réguliers. En effet, les structures auto-similaires des représentations observées sont caractérisables par des L-systèmes, construitàpartir d'automates finis.

Grâce aux automates probabilistes, nous avons également permis de quantifier l'influence d'un motif particulier dans une représentation. En effet, les motifs présents traduisent la présence de langages exceptionnels dans la séquence. Ces caractéristiques statistiques se traduisent aisément par des automates probabilistes.

Nous avons ainsi mis au point une technique de recherche de motifs dans les séquences par filtrage successifs. En effet, les automates probabilistes permettent de modéliser précisément l'influence d'un motif dans la séquence. En filtrant la représentation par le modèle statistique décrit par l'automate, on le supprime de la représentation. Il est ainsi possible, par filtrages successifs par des modèles de plus en plus complexes, d'aprocher de plus en plus près le modèle statistique de la séquence.

Cette caractérisation des motifs permet aussi de discriminer plus efficacement les séquences. En effet, jusqu'à présent, pour discriminer les séquence à l'aide du jeu du chaos, on utilisait la distribution de fréquences des mots de taille 8. En utilisant des automates modélisants des motifs importants, on peut discriminer des séquences sur des aspects précis plutôt que sur la totalité de la distribution de fréquences, réduisant du même coup, la quantité de variables à analyser.

Nous avons également appliqué le jeu du chaos dans la recherche de variations de distribution de fréquence le long de la séquence. Ainsi, la représentation par jeu du chaos de morceaux de la séquence permet de visualiser la présence de *crisprs*. Cette technique fournis une méthode complémentaire de visualisation aux techniques de recherches de *crisprs* en permettant de les visualiser plus facilement.

Enfin, le jeu du chaos a permis de découvrir une nouvelle symétrie statistique dans les séquences d'ADN. En effet, grace à la représentation d'une séquence et celle de son inverse, on a pu voir que chaque mot avait la même fréquence que son complémentaire palindromique. Ce résultat est intéressant dans la mesure où il n'a pas encore été montré.

Ainsi, on sait maintenant pourquoi les représentations par jeu du chaos contiennent des structures auto-similaires et on dispose de techniques pour les caractériser et les filtrer. Cette technique de représentation des séquences est aussi très riche en applications, comme la discrimination de séquences et la recherche de modèles statistiques.

Bibliographie

[1] http ://www.cns.fr/externe/francais/questions/. Site au sujet du projet Génome.

[2] R. Mansilla, N. Del Castillo, T. Govezensky, P. Miramontes, M. Jose, and G. Coch. Long-range correlation in the whole human genome. *eprint arXiv :q-bio/0402043*, February 2004.

[3] Pedro Bernaola-Galván, Ramón Román-Roldán, and josé L. Oliver. Compositional segmentation and long-range fractal correlation in dna sequences. 1996.

[4] Nestor Oiwa and James A. Glazier. Self-similar mitochondrial dna. 2004.

[5] Marcelo C. Pinton and Günteher J. L. Gerhardt. Finding periodicities in dna sequences with a wavelet technique.

[6] Yasubumi Sakakibara, Michael Brown, Richard hughey, I. Saira Mian, Kimmen Sjölander, Rebecca C. underwood, and David Haussler. Stochastic context-free grammar for trna modeling. 1994.

[7] David B. Searls. The language of genes. 2002.

[8] H. Joel Jeffrey. Chaos game representation of gene structure. 1990.

[9] Peter Tiño. Spacial representation of symbolic sequences through iterativ functgion systems. 1999.

[10] Kenneth Falconer. *Fractal Geometry*. John Wiley, 2003.

[11] http ://www.mathcurve.com/fractals/lsysteme/lsysteme.shtml. Site décrivant les L-systèmes.

[12] Patrick J. Deshavanne, Alain giron, Joseph vilain, Guillaume Fagot, and Bernard Fertil. Genomic signature : characterization and classification of species assessed by chaos game representation of sequences. 1999.

[13] Jérôme Callut and Pierre dupont. Inducing hidden markov models to model long-term dependencies.

[14] O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005.

[15] http ://fr.wikipedia.org/wiki/cha%c3%aene_de_markov. Encyclopédie Wikipedia, définition des chaînes de Markov.

[16] Natsuhiro Ichinose, Tetsushi Yada, and Toshihisa Takagi. Quadtree representation of dna sequences. 2001.

[17] Bernard Fertil, Matthieu Massin, Sylvain Lespinats, Caroline Devic, Philippe Dumee, and Alain Giron. Genstyle : exploration and analysis of dna sequences with genomic signature. 2005.

[18] Gorban Alexander, Popova Tatyana, and Zinovyev Andrey. Four basic symmetry types in the universal 7-cluster structure of 143 complete bacterial genomic sequences. 2004.

[19] R. Jansen, J. D. A. Van Embden, W. Gaastra, and L. M. Schouls. Identification of genes that are associated with dna repeats in prokaryotes. *Mol Microbiol*, 43(6) :1565–1575, 2002.

[20] Durand Patrick, Mahé Frédéric, Valin Anne-Sophie, and Nicolas Jacques. Rr-5798 - pyramid diagram : visualizing the organization of repetitive sequences in genome. 2005.

[21] Képès F. Periodic transcriptional organization of the *E. coli* genome *http ://stat.genopole.cnrs.fr/%7Ekepes/kepes2004a.pdf*. 2004.

[22] Marina A. Makarova and Michael G.Sadovsky. New symmetry in nucleotide sequence. 2005.

[23] F. Thollard and A. Clark. Apprentissage d'automates probabilistes déterministes. In *Conférence d'Apprentissage*, 2002.