Mémoire d'Habilitation à Diriger des Recherches

soutenu le 10 décembre 2001
à l'université Paris-Sud

Spécialité : Informatique

Structures Aléatoires
Modèles
et Analyse des Génomes

Alain Denise

deuxième tirage, mars 2002.

Remerciements.

Philipp Bucher, Maxime Crochemore, Bernard Prum et Volker Strehl m'ont honoré en acceptant d'être les rapporteurs de ce mémoire. Je les en remercie vivement.

Je n'ai d'abord connu Philipp Bucher qu'à travers ses travaux. Son avis de biologiste spécialiste de la modélisation et de la prédiction de l'information génétique m'était indispensable. J'ai découvert un homme ouvert et chaleureux, avec qui j'espère bien continuer à discuter de génomique.

Depuis longtemps, je connais et respecte Volker Strehl pour son humanité et sa rigueur scientifique ; je tenais à ce qu'il évalue mes travaux, et suis profondément heureux de l'intérêt qu'il leur manifeste.

Maxime Crochemore et Bernard Prum sont parmi les personnes qui animent et font progresser la communauté qui travaille à l'interface entre Mathématiques, Informatique et Biologie en France, et leur avis sur mes travaux dans ce domaine m'était très cher. Tous deux m'ont chaleureusement accueilli au sein de cette communauté, et leurs encouragements me touchent particulièrement.

Je suis très reconnaissant à Serge Dulucq, Christine Froidevaux, Dominique Gouyou-Beauchamps et Michel Termier de m'avoir fait l'honneur de participer au jury.

Dès le début de ma thèse au LaBRI, j'ai pu découvrir les qualités scientifiques et humaines de Serge Dulucq, qui m'ont très vite donné envie de collaborer avec lui. J'ai dû attendre quelques années, mais c'est maintenant presque chose faite puisque nous avons un projet commun en génomique.

Christine Froidevaux est l'une des premières personnes converties à la bioinformatique au LRI. Son engagement dans la collaboration pluridisciplinaire avec les biologistes d'Orsay est profond et fécond. Je lui sais gré pour les responsabilités qu'elle a accepté de prendre dans ce cadre, et suis très honoré qu'elle ait été présidente du jury.

Dominique Gouyou-Beauchamps était déjà membre du jury de ma thèse. Je suis heureux qu'il ait à nouveau accepté de jouer ce rôle. Il m'a très chaleureusement accueilli dans son équipe au LRI en 1995. Depuis lors, j'ai eu largement le temps d'apprécier combien est précieuse son amitié et profonde sa rigueur scientifique. Je ne compte plus les occasions où j'ai fait appel à son jugement et où il m'a guidé et encouragé dans la voie que je choisissais.

Michel Termier porte l'entière responsabilité de ma "mutation" vers la bioinformatique. C'est lui qui m'a patiemment formé à la Biologie et qui m'a convaincu de la nécessité de se plonger véritablement dans "l'autre discipline" pour mettre en œuvre un projet de recherche pluridisciplinaire. Il est entièrement à l'origine de la dynamique qui existe à Orsay entre biologistes, informaticiens et mathématiciens, tant en recherche qu'en enseignement.

Mon parcours d'enseignant-chercheur est jalonné de rencontres qui m'ont enrichi, aussi bien du point de vue humain que scientifique. Je voudrais remercier encore quelques personnes sans qui ce mémoire ne serait pas ce qu'il est. Merci donc à ...

Michel Duguet, pour m'avoir accueilli à l'IGM et y avoir favorisé le développement de la bioinformatique. Il a toujours été disponible pour m'écouter et m'encourager malgré le peu de temps que lui laissaient ses responsabilités.

Rachid Gherbi. Lors de mon arrivée à Orsay, j'ai trouvé en Rachid un coéquipier avec qui j'ai eu une entente exceptionnelle pour l'organisation des enseignements dont nous étions coresponsables. A mon grand plaisir, c'est ensemble que nous avons commencé à travailler en bioinformatique et notre collaboration s'étend à la recherche.

Jean-Pierre Rousset. Jean-Pierre est un biologiste "humide" qui a, sans hésiter, franchi le pas vers la bioinformatique. C'est à lui que nous devons une collaboration exemplaire entre l'IGM et le LRI, allant de la modélisation formelle à l'expérimentation biologique. Jean-Pierre m'a beaucoup appris en Biologie moléculaire. Sa bonne humeur et ses qualités humaines font de nos réunions de travail des forts moments de convivialité.

Claire Toffano. Sa double culture de biologiste et d'informaticienne, associée à sa gentillesse, ont permis au petit groupe de départ de trouver rapidement un langage commun et des problématiques à explorer ensemble.

Robert Cori et Philippe Flajolet, qui ont toujours suivi mes travaux avec bienveillance et m'ont, à plusieurs reprises, aidé à aller plus avant dans mes recherches.

Mireille Régnier, qui m'a initié à la statistique des génomes et m'a offert de collaborer avec elle dans ce domaine.

Marie-Claude Gaudel. Grâce à elle, j'ai le plaisir de pratiquer une autre forme d'interdisciplinarité, entre génie logiciel et combinatoire.

Paul Zimmermann, avec qui j'ai eu le bonheur de collaborer, et qui a pris le temps de lire et annoter de façon extrêmement constructive une bonne partie de ce mémoire, tout comme il l'avait fait pour ma thèse.

Sylvie Corteel, qui conçoit des bijections plus vite que son ombre, et avec qui les séances de travail sont joyeuses et animées.

Jean-Paul Allouche. J'ai la chance d'avoir un voisin de bureau cordial, stimulant, et de profiter de ses éclaircissements sur bien des points des mathématiques.

Dominique Barth, mon ami de longue date, qui, après m'avoir attiré un temps vers la combinatoire des réseaux d'interconnexion, se laisse à son tour volontiers séduire par la bioinformatique.

les membres des équipes BIG et GMT de l'IGM, qui m'ont accueilli à bras ouverts : Laure Bidou, Guillemette Duchateau-Nguyen, Maryse Goron, Isabelle Hatin, Florence Mougel, Olivier Namy.

les jeunes chercheurs de mon entourage qui se plongent avec enthousiasme dans la recherche interdisciplinaire : David Abergel, Michaël Bekaert, Jean-Paul Forest, Sandrine Gouraud, Yann Ponty, Romain Rivière, Mathias Vandenbogaert.

mes initiateurs en combinatoire : Jean-Guy Penaud, qui m'a offert un très beau sujet de thèse ; Maylis Delest et Jean-Marc Fédou qui ont suivi mes tout premiers pas en combinatoire ; Rodica Simion qui m'a tant appris lors de notre collaboration.

mes (autres) coauteurs : Cyril Banderier, Mireille Bousquet-Mélou, Christoph Dürr, Isabelle Dutour, Danièle Gardy, Fouad Ibn-Majdoub-Hassani, Bruno Marre, Olivier Roques, Nadine Rouillon, Mario Valencia-Pabon, Marcio Vasconcellos, Dominic Welsh.

Table des matières

Introduction.

Depuis le début de ma thèse en 1991 jusqu'il y a trois ans environ, j'ai travaillé en combinatoire énumérative, principalement dans le domaine de la génération aléatoire de structures combinatoires. En 1998, j'eus la chance de rencontrer Michel Termier –par hasard : nous enseignions dans les mêmes locaux. Il me parla de génomes et me confia son intérêt pour l'informatique, autant formelle que pratique. De mon côté, j'avais toujours été fasciné par les progrès de la biologie moléculaire, sans toutefois y comprendre grand chose. C'était l'occasion de se lancer. Bientôt, nous formâmes un petit groupe de travail avec Rachid Gherbi et Claire Toffano. L'idée était d'aborder les problème d'analyse des génomes dans un esprit totalement pluridisciplinaire. Mais il fallait d'abord que chacun d'entre nous enseigne aux autres une partie de sa propre discipline. Je savais, bien sûr, qu'il existait des liens forts entre informatique et étude des génomes. Je découvris, en même temps que les notions fondamentales de la biologie moléculaire, d'autres liens que je ne soupçonnais pas entre ces deux disciplines. Je me rendis compte avec étonnement que la génération et l'étude des séquences aléatoires sont de quelque importance dans la recherche en génomique. Ainsi, je me retrouvais (un peu) sur un terrain connu. J'ai maintenant la chance de travailler dans un domaine à la frontière entre informatique et biologie, sur des problématiques par lesquelles les deux disciplines s'enrichissent mutuellement.

Le premier chapitre de ce mémoire est consacré exclusivement à l'algorithmique de la génération aléatoire d'objets combinatoires. Il s'agit d'une synthèse de l'état de l'art, dans laquelle mes propres contributions sont mises en relief. La grande majorité des travaux concernent la génération *uniforme*, c'est-à-dire équiprobable. L'influence de la bioinformatique sur mes recherches dans le domaine se traduit, on le verra, par l'étude de la génération *non uniforme* de mots de langages. La fin du chapitre est consacrée à un travail qui lie la génération aléatoire au test statistique de logiciels.

Les séquences aléatoires sont presque universellement présentes en analyse des séquences. Elles sont principalement utilisées pour représenter le «bruit de fond» des génomes, dont ce qui constitue la véritable information génétique doit logiquement se différencier. Elles doivent obéir à des modèles (que j'appellerai *modèles exploratoires*) qui reflètent des propriétés bien choisies des séquences réelles. Il n'y a pas eu jusqu'à présent de réflexion globale sur la conception et l'utilisation de ces modèles. C'est pourquoi j'ai choisi de présenter dans le second chapitre une synthèse des modèles existants et d'entamer une réflexion sur leurs raisons d'être. J'y décris mes travaux de conception et de mise en œuvre de modèles plus complexes que ceux existant actuellement, notamment pour mettre en évidence des propriétés plus fines des séquences biologiques. La plupart des travaux personnels que je présente sont en cours ; aussi l'exposé est plus riche en perspectives qu'en résultats ! Le chapitre se termine par une présentation de deux études auxquels je participe, qui concernent l'analyse des séquences selon des approches pluridisciplinaires intégrées. Ce sont des travaux éminemment collaboratifs, qui ne sont possibles que parce que d'autres personnes –biologistes, informaticiens, mathématiciens– se sont jointes au petit groupe de départ, ont fait l'effort d'apprendre les bases (voire plus) des autres disciplines et ont apporté leurs savoir-faire et leurs problématiques. Mon but est que la partie bioinformatique du mémoire soit compréhensible par des informaticiens ou mathématiciens non familiers de la génomique. J'ai donc jugé nécessaire de consacrer une bonne part du second chapitre aux notions fondamentales de biologie moléculaire, ainsi qu'aux méthodes et modèles utilisés dans l'analyse *in silico*[1] des séquences.

Mes autres travaux, qui relèvent de la combinatoire énumérative, sont brièvement résumés dans une annexe en fin de rapport.

1. c'est-à-dire par ordinateur.

Chapitre 1

Génération aléatoire de structures combinatoires.

1.1 Introduction.

Depuis que l'ordinateur existe, il est utilisé pour modéliser et simuler des phénomènes du monde physique. Beaucoup de ces phénomènes (la trajectoire d'une molécule, une mutation dans un chromosome, les fluctuations du cours de la Bourse, ...) sont sujets à ce qu'on appelle le hasard. Ainsi les ordinateurs doivent-ils être capables de mimer ce hasard. Herbert Wilf, le père de la génération aléatoire de structures combinatoires, explique dans [185] que c'est son travail sur la simulation de trajectoires de particules à l'intérieur de réacteurs nucléaires, dans les années soixante, qui l'a mené à la génération aléatoire. Le travail de pionnier qu'il a effectué dans le domaine (voir [184] et [132], avec Albert Nijenhuis) concerne la génération aléatoire *uniforme* ou *équiprobable*. Le problème est le suivant : étant donné un ensemble E d'objets combinatoires et un entier n, soit E_n l'ensemble des éléments de E de taille n. Il s'agit d'engendrer un ou plusieurs éléments de E_n de sorte que tous aient la même probabilité d'apparaître.

La validation de modèles en est une application particulièrement importante. Par exemple, dans [179], Bernard Van Cutsem décrit en quoi la génération aléatoire de structures combinatoires est utile en classification pour déterminer si une hiérarchie donnée est pertinente. Schématiquement, il s'agit de comparer les propriétés de la hiérarchie en question à celles de hiérarchies aléatoires. Si celles-ci sont suffisamment similaires, la hiérarchie peut être considérée comme le résultat d'un processus aléatoire, et donc non informative. Ce type d'approche est classique lorsque l'on doit confronter une hypothèse à «l'hypothèse nulle» matérialisée par les structures aléatoires. Nous en verrons des illustrations dans le cadre de la génomique au chapitre 2.

En informatique, des graphes aléatoires sont utilisés pour modéliser des réseaux de communications sur lesquels on étudie le comportement d'algorithmes de routage. Le test statistique de programmes consiste à injecter dans les programmes des données aléatoires afin d'y identifier les erreurs. De façon similaire, on évalue empiriquement la complexité d'algorithmes en leur fournissant des données aléatoires en entrée.

L'objectif général de la combinatoire est d'accéder à une connaissance profonde des propriétés des structures discrètes. Dans cette discipline, l'étude de la génération aléatoire se justifie naturellement par le fait que savoir engendrer des objets est, en quelque sorte, une preuve qu'on en a une connaissance intime. D'autre part, la visualisation ou la mesure de paramètres sur des objets aléatoires peuvent induire la découverte de propriétés nouvelles de ces objets.

La génération aléatoire est aussi un moyen de mesurer expérimentalement des quantités. Il en est ainsi de la méthode de Monte-Carlo, dont l'illustration la plus simple (l'estimation d'une aire) est bien connue. Les liens forts qui existent entre dénombrement et génération aléatoire [112] permettent d'utiliser des procédés de génération pour dénombrer approximativement des objets combinatoires lorsque leur dénombrement exact ne peut être envisagé [110].

Les simulations exigent généralement d'engendrer un grand nombre d'objets de grande taille. C'est pourquoi l'efficacité des algorithmes est un problème crucial. Ainsi, pour engendrer des structures combinatoires de taille n, un algorithme en $\Theta(n^2)$ est «limite»; un algorithme en $\Theta(n^3)$ est inacceptable, sauf cas très particulier. Nous verrons que la plupart des travaux s'attachent à améliorer la complexité des méthodes génériques, ou bien à proposer des méthodes *ad hoc* très efficaces pour des classes particulières d'objets. Deux mesures de complexité sont utilisées dans ce document. La *complexité arithmétique* repose sur l'hypothèse que la place occupée par un nombre est constante et qu'une opération arithmétique élémentaire (addition, multiplication) s'effectue en temps constant. Cette mesure est réaliste si les nombres mis en jeu sont bornés. En revanche, s'ils sont très grands, elle ne reflète pas le temps et la mémoire réellement utilisés par l'algorithme. Dans ce cas, meilleure est la *complexité logarithmique* (appelée aussi *complexité binaire*), qui repose sur les hypothèses suivantes : la place occupée par un nombre k est de l'ordre de $\log k$; une opération arithmétique simple s'effectue en temps $O(M(k))$, où $M(k)$ dépend de l'algorithme utilisé pour l'opération (ce sera évoqué en section 1.2.3).

Il existe trois principaux types d'approches pour la génération aléatoire : la méthode récursive, les méthodes à rejet et les méthodes à base de chaînes de Markov. Un outil complémentaire est le *codage*, qui peut être utilisé conjointement aux trois méthodes : pour engendrer des objets d'un ensemble E, on établit une bijection avec ceux d'un ensemble F que l'on sait mieux manipuler. Ceci fait, on engendre des objets de F, puis on les décode pour obtenir les objets de E correspondants. Généralement, on code des objets par des mots (les mots sont des structures linéaires que l'on sait bien appréhender en général), mais ce n'est pas systématique. Nous verrons des exemples de codages tout au long du chapitre.

1.2 L'approche récursive.

1.2.1 Génération uniforme de structures décomposables.

Premières formalisations.

Dans son article fondateur de 1977 [184], Herbert Wilf considère une famille de structures combinatoires dont la construction peut être représentée par un chemin dans un graphe orienté acyclique. L'exemple des sous-ensembles de cardinal k d'un ensemble $E = \{e_1,e_2,\ldots,e_n\}$ illustre très simplement le principe. Les étapes successives de la construction d'un tel sous-ensemble peuvent être représentées par un chemin dans le plan discret de longueur n et de hauteur k qui n'emprunte que des pas $s_1 = (+1,+1)$ et $s_0 = (+1,0)$. Lorsqu'on est au point (i,j), le choix du pas s_1 détermine l'appartenance de l'élément e_i au sous-ensemble ; le choix de s_0 détermine sa non appartenance. Notons $\mathcal{C}_{n,k}$ l'ensemble de ces chemins et convenons d'appeler respectivement longueur et hauteur les entiers n et k. L'algorithme de génération uniforme tire avantage de la structure récursive des chemins. Tout chemin de $\mathcal{C}_{i,j}$ est soit un pas s_0 suivi d'un chemin de $\mathcal{C}_{i-1,j}$, soit un pas s_1 suivi d'un chemin de $\mathcal{C}_{i-1,j-1}$. De manière plus concise : $\mathcal{C}_{i,j} = p_0.\mathcal{C}_{i-1,j} + p_1.\mathcal{C}_{i-1,j-1}$, où le signe «+» désigne l'union disjointe. A chaque étape, on va donc choisir d'engendrer un pas s_0 avec probabilité $|\mathcal{C}_{i-1,j}|/|\mathcal{C}_{i,j}| = (i-j)/i$ et un pas p_1 avec probabilité $|\mathcal{C}_{i-1,j-1}|/|\mathcal{C}_{i,j}| = j/i$ (où $|\mathcal{C}_{i,j}|$ désigne le cardinal de $\mathcal{C}_{i,j}$).

Les principes fondamentaux de la méthode sont posés dans l'article de Wilf. Le point de départ est une description récursive non ambiguë de l'ensemble dans lequel on veut tirer des objets combinatoires. Cette description détermine des choix à effectuer pour engendrer pas à pas un objet. Les probabilités de choix se calculent en fonction des cardinalités des ensembles mis en jeu dans la description récursive. Cependant, le propos est restreint aux structures combinatoires qui sont en bijection avec des chemins du plan particuliers. (Notons qu'Arnold et Sleep, en 1980, «redécouvriront» la méthode de Wilf dans le cas particulier de la génération des chemins de Dyck [10] [1].) En 1979, le principe est transposé à des ensembles plus généraux de structures combinatoires dans l'ouvrage «Combinatorial Algorithms» de Nijenhuis et Wilf [132, pp. 78–87]. On y considère des objets combinatoires qui peuvent s'exprimer de façon unique à partir d'un ensemble (éventuellement infini) d'objets primitifs et d'un opérateur de composition $\otimes$, commutatif et associatif. Les partitions d'entiers constituent un exemple canonique : toute partition peut s'écrire sous la forme $p_1^{\mu_1} \otimes p_2^{\mu_2} \otimes \cdots p_l^{\mu_l}$, où les p_i, deux à deux différents, sont les partitions réduites à un seul entier. Les auteurs notent que les objets combinatoires ainsi définis sont en fait les multi-ensembles d'objets primitifs. L'algorithme qu'ils proposent permet donc d'engendrer uniformément des multi-ensembles d'objets combinatoires à condition de savoir engendrer uniformément les objets en question (considérés comme des objets primitifs). En particulier, la méthode s'applique aux arbres enracinés non plans (arbres enracinés dans lesquels l'ordre des fils d'un sommet n'est pas pris en compte) : un tel arbre est un multi-ensemble d'arbres de la même famille, auquel on ajoute un sommet que l'on relie à toutes leurs racines ; ce sommet devient la nouvelle racine. Ainsi, le terme de *décomposition récursive* prend tout son sens. En 1979, Tinhofer applique cette approche à la génération de certaines familles de graphes [172, 174]. Hickey et Cohen [105] proposent en 1983 une méthode de génération de mots de langages algébriques fondée sur un principe proche. La décomposition récursive est naturellement fournie par une grammaire du langage. Nous reviendrons sur cette méthode dans la section 1.2.2 consacrée aux langages algébriques.

Spécifications combinatoires et spécifications standard.

C'est en 1994 que Flajolet, Zimmermann et Van Cutsem [86] publient un schéma général de composition de structures combinatoires qui englobe en un même formalisme les travaux précédents. Ce schéma repose sur la notion de *spécification combinatoire*. Dans ce formalisme, les objets primitifs sont l'objet «vide» (de taille 0) noté 1, et un ensemble fini d'*atomes* de taille 1. (Dans la suite, le symbole Z désignera un atome générique). Cinq opérateurs permettent de définir récursivement des ensembles d'objets combinatoires à partir d'autres ensembles et des deux types d'objets primitifs :

– L'union disjointe (+).

1. Un chemin de Dyck est un chemin composé uniquement de pas $(+1,+1)$ et $(+1,-1)$, qui débute au point (0,0) et se termine en un point $(n,0)$. Les chemins de Dyck sont en bijection avec les systèmes de parenthèses bien formés, avec les arbres binaires et bien d'autres structures.

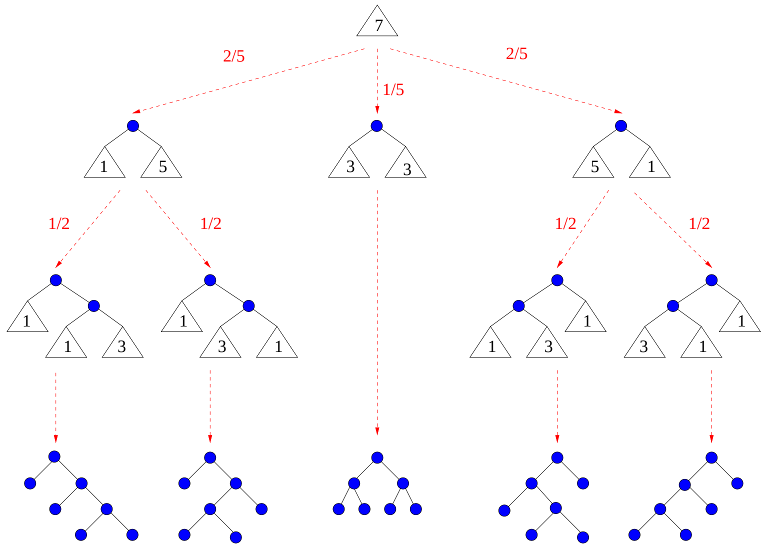


Fig. 1.1 – *Illustration de la méthode récursive. Un arbre binaire complet est un arbre qui est soit réduit à un sommet, soit formé d'un sommet racine, d'un sous-arbre gauche et d'un sous-arbre droit. La figure représente toutes les exécutions possibles d'un algorithme de génération d'un arbre à 7 sommets. (Cet algorithme a été choisi pour sa simplicité ; il en existe des bien plus efficaces [10, 15, 149].) Le nombre à l'intérieur de chaque triangle est le nombre de sommets que devra contenir le sous-arbre. Les probabilités des choix sont indiquées sur les arêtes.*

- Le produit (non commutatif) : $A \cdot B$ est l'ensemble des couples composés d'un élément de A et d'un élément de B.
- La séquence, l'ensemble et le cycle : `sequence`(A) (resp. `set`(A), `cycle`(A)) désigne l'ensemble des suites (resp. ensembles, cycles) finies d'éléments de A. Ces opérateurs peuvent être accompagnés d'un argument qui fixe une condition sur la cardinalité des suites (resp. ensembles, cycles).

Dans l'article initial, les objets sont *étiquetés* : à chaque atome d'un objet de taille n est associé un entier. L'étiquetage doit être canonique : si l'objet est de taille n, les étiquettes sont entre 1 et n et deux atomes différents ont toujours des étiquettes différentes. Cela implique que, lors des quatre dernières opérations ci-dessus, un réétiquetage des objets est effectué. Ainsi, par exemple, le produit n'est pas un produit cartésien, comme illustré ci-après :

$$\{\bullet_1\} \cdot \{\overset{\bullet 1}{\underset{\bullet 2}{|}}\} \;=\; \{(\bullet_1, \overset{\bullet 2}{\underset{\bullet 3}{|}})\;,\;(\bullet_2, \overset{\bullet 1}{\underset{\bullet 3}{|}})\;,\;(\bullet_3, \overset{\bullet 1}{\underset{\bullet 2}{|}})\;,\;\}.$$

Dans un travail ultérieur [87], les mêmes auteurs adaptent leur formalisme aux objets non étiquetés. Dans ce cas, les opérateurs produit, séquence et cycle ont leur sémantique usuelle. En particulier, le produit est bien un produit cartésien :

$$\{\bullet\} \cdot \{\overset{\bullet}{\underset{\bullet}{|}}\} \;=\; \{\bullet\} \times \{\overset{\bullet}{\underset{\bullet}{|}}\} \;=\; \{(\bullet, \overset{\bullet}{\underset{\bullet}{|}})\}.$$

En revanche, l'opérateur `set` pour les objets non étiquetés ne construit pas des ensembles, mais des multi-ensembles :

$$\texttt{set}(\{\bullet, \overset{\bullet}{\underset{\bullet}{|}}\}) \;=\; \{\emptyset, \{\bullet\}, \{\overset{\bullet}{\underset{\bullet}{|}}\}, \{\bullet, \bullet\}, \{\bullet, \overset{\bullet}{\underset{\bullet}{|}}\}, \{\overset{\bullet}{\underset{\bullet}{|}}, \overset{\bullet}{\underset{\bullet}{|}}\}, \{\bullet, \bullet, \bullet\}, \{\bullet, \bullet, \overset{\bullet}{\underset{\bullet}{|}}\}, \{\bullet, \overset{\bullet}{\underset{\bullet}{|}}, \overset{\bullet}{\underset{\bullet}{|}}\}, \cdots\}.$$

Structures étiquetées :	Permutations :	$P = \texttt{sequence}(Z)$
		ou $P = \texttt{set}(\texttt{cycle}(Z))$
	Partitions d'ensembles :	$P = \texttt{set}(\texttt{set}(Z, \text{card} \geq 1))$
	Surjections :	$S = \texttt{sequence}(\texttt{set}(Z, \text{card} \geq 1))$
Structures non étiquetées :	Partitions d'entiers :	$F = \texttt{set}(\texttt{sequence}(Z, \text{card} \geq 1))$
	Compositions d'entiers :	$F = \texttt{set}(\texttt{set}(Z, \text{card} \geq 1))$
	Chemins de Dyck :	$D = 1 \ + \ Z \cdot D \cdot \bar{Z} \cdot D$
Indifférent :	Arbres binaires complets	
	atomes = feuilles :	$B = Z \ + \ B \cdot B$
	atomes = sommets :	$B = Z \ + \ Z \cdot B \cdot B$
	Arbres généraux	
	plans :	$A = Z \cdot \texttt{sequence}(A)$
	non plans :	$A = Z \cdot \texttt{set}(A)$

FIG. 1.2 – *Quelques spécifications combinatoires.*

Définition 1 (Flajolet, Zimmermann, Van Cutsem 1994 [86]) *Soit $T = \{T_0,T_1,\ldots,T_m\}$ une famille de $m+1$ ensembles d'objets combinatoires (étiquetés ou non). Une* spécification combinatoire *de T est un ensemble de $m+1$ équations telles que la ième (pour tout $0 \leq i \leq m$) s'écrit $T_i = \Psi_i(T_0,T_1,\ldots,T_m)$, où Ψ_i est une combinaison des cinq opérateurs définis ci-dessus appliquée aux T_i, à l'objet vide et aux atomes.*

Les multi-ensembles considérés par Nijenhuis et Wilf peuvent bien évidemment être décrits de la sorte, ainsi que les langages algébriques. La figure 1.2 présente quelques exemples de spécifications pour des objets combinatoires classiques.

Il existe un lien direct entre une spécification combinatoire et la série génératrice des objets qu'elle définit : des règles simples permettent de déduire la seconde de la première. En revanche, pour aboutir à des algorithmes de génération aléatoire efficace, il est nécessaire de transformer la spécification en *spécification standard.* En voici la définition en ce qui concerne les objets étiquetés.

Définition 2 (Flajolet, Zimmermann, Van Cutsem 1994 [86]) *Soit $T = \{T_0,T_1,\ldots,T_m\}$ une famille de $m+1$ ensembles d'objets combinatoires étiquetés. Une* spécification standard *de T est un ensemble de $m+1$ équations telles que la ième (pour tout $0 \leq i \leq m$) s'écrit $T_i = 1$ ou $T_i = Z$ ou $T_i = U_j + U_k$ ou $T_i = U_j \cdot U_k$ ou $\Theta T_i = U_j \cdot U_k$. Chaque U_j appartient à $\{1,Z,T_0,\ldots,T_m,\Theta T_0,\ldots,\Theta T_m\}$. Le symbole Θ désigne l'opérateur de* pointage *:*

$$\Theta A \quad = \quad \bigcup_{n=1}^{\infty} (\mathcal{A}_n \times \{1,2,\ldots,n\})$$

où $\mathcal{A}_n$ est l'ensemble des objets de A de taille n.

Nous ne détaillerons pas les spécifications standard d'objets non étiquetés. La seule différence notable est la présence d'un opérateur supplémentaire Δ, appelé diagonale.

Théorème 3 (Flajolet, Zimmermann, Van Cutsem 1994 [86]) *Tout ensemble qui admet une spécification combinatoire admet une spécification standard.*

La preuve du théorème est constructive : elle consiste en un ensemble de règles de traduction d'une spécification en spécification standard équivalente. La traduction de l'union et du produit ne pose pas de problème. Les autres opérateurs sont sujets aux règles suivantes :

$$B = \mathtt{sequence}(A) \quad \to \quad B = 1 + A \cdot B,$$

selon le schéma classique de décomposition récursive d'une suite ;

$$\begin{aligned} B = \mathtt{set}(A) \quad &\to \quad \Theta B = B \cdot \Theta A, \\ B = \mathtt{cycle}(A) \quad &\to \quad \Theta B = \mathtt{sequence}(A) \cdot \Theta A, \end{aligned}$$

car pointer un atome dans un ensemble ou un cycle revient à singulariser l'objet portant l'atome en question. Nous ne détaillerons pas le cas non étiqueté dans lequel seule la traduction des opérateurs `set` (multi-ensemble) et `cycle` est modifiée ; elle fait appel à l'opérateur Δ.

Algorithmes pour la génération.

L'algorithme de génération aléatoire se déduit de la spécification standard. Soit n la taille des objets à engendrer. La première étape est une étape de dénombrement : il s'agit de calculer, pour tout ensemble C intervenant dans la spécification standard et pour tout $0 \leq i \leq n$, le nombre c_i d'objets de taille i de C. Il s'agit du coefficient de degré i de la série génératrice

$$C(z) = \sum_{k \geq 0} c_k z^k.$$

Pour ce faire, on déduit directement de la spécification standard les relations de récurrence suivantes :

$$\begin{aligned} C = 1 \;\Rightarrow\; c_0 = 1 \;; &\qquad C = Z \;\Rightarrow\; c_1 = 1 \;; \\ C = A + B \;\Rightarrow\; c_n = a_n + b_n \;; &\qquad C = A \cdot B \;\Rightarrow\; c_n = \sum_{k=0}^{n} \binom{n}{k} a_k b_{n-k} \;; \\ \Theta C = A \cdot B \;\Rightarrow\; c_n = \frac{1}{n} \sum_{k=0}^{n} \binom{n}{k} a_k b_{n-k} \;; &\qquad C = \Theta A \;\Rightarrow\; c_n = n a_n. \end{aligned}$$

Le nombre d'opérations arithmétiques effectuées au cours de l'étape de dénombrement est clairement en $O(n^2)$ (et peut être en $\Theta(n^2)$ si la spécification standard contient au moins un produit). Certaines classes d'objets possèdent une série génératrice *holonome* : il existe (et on sait calculer) une formule de récurrence linéaire à coefficients polynomiaux qui donne les termes de la série génératrice. Ceci implique que les termes jusqu'à l'ordre n peuvent être calculés en $O(n)$ opérations arithmétiques. Dans le cas général comme dans le cas holonome, $O(n)$ entiers doivent être stockés.

La phase de dénombrement n'est effectuée qu'une fois, quel que soit le nombre d'objets de taille n (ou inférieure) que l'on désire engendrer. Le processus de génération s'effectue de façon récursive. A chaque ensemble C d'objets représenté dans la spécification standard, on associe une des procédures ci-dessous. Les procédures de pointage et de dépointage, nécessaires pour certaines étapes (notamment le cas $\Theta C = A \cdot B$) ne sont pas détaillées car triviales : engendrer un élément pointé consiste à engendrer un élément non pointé et lui associer un entier entre 1 et n.

```
Case: C = 1.
  gC := procedure(n: integer);
    if n = 0 then Return(1)
  end.

Case: C = Z.
  gC := procedure(n: integer);
    if n = 1 then Return(Z)
  end.

Case: C = A + B.
  gC := procedure(n: integer);
    U:=Uniform([0,1]);
    if U < a_n/c_n
      then Return(gA(n))
      else Return(gB(n))
  end.

Case: C = A · B.
  gC := procedure(n: integer);
    U:=Uniform([0,1]);
    k := 0;
    S := a_0 b_n/c_n;
    while U > S do
      k := k + 1;
      S := S + a_k b_{n-k}/c_n;
    Return([gA(k),gB(n - k)])
  end.
```

Le cas $C = A+B$ implique un choix binaire : on engendre un élément de A avec probabilité a_n/c_n ou un élément de B avec probabilité b_n/c_n. Le cas $C = A \cdot B$ nécessite de déterminer les tailles respectives des objets de A et de B à engendrer. L'objet de A sera de taille k avec probabilité $a_k b_{n-k}/c_n$ (et celui de B sera de taille $n - k$). En raison de la boucle du cas $C = A \cdot B$, il est clair qu'il existe des spécifications standard pour lesquelles l'algorithme nécessite $\Theta(n^2)$ opérations dans le pire des cas, et aussi en moyenne. Il suffit de considérer une spécification qui ne contient que des règles de type $C = A \cdot B$ et $C = Z$. Flajolet, Zimmermann et Van Cutsem décrivent un moyen particulièrement étonnant d'améliorer très sensiblement cette complexité en temps. Dans le cas $C = A \cdot B$, soit K la variable aléatoire qui représente la taille de l'élément de A, et $\pi_{n,k} = \Pr(K = k) = a_k b_{n-k}/c_n$. L'algorithme présenté ci-dessus ajoute successivement ces probabilités à S dans l'ordre $\pi_{n,0},\pi_{n,1},\pi_{n,2},\ldots$ Considérons maintenant un algorithme qui effectue le même traitement, mais dans l'ordre suivant : $\pi_{n,0},\pi_{n,n},\pi_{n,1},\pi_{n,n-1},\pi_{n,2},\pi_{n,n-2},\ldots$ Cette variante est appelée «boustrophédon»[2] dans [86]. Cette seule modification donne une complexité arithmétique en $O(n \log n)$ dans le pire des cas, au lieu de $O(n^2)$! Le principe de la preuve est extrêmement simple : soit $f(n)$ la complexité au pire. Elle satisfait une récurrence de type

$$f(n) = \max_{0 \leq k \leq n} (f(k) + f(n-k) + 2\min(k,n-k)).$$

dont la solution est en $O(n \log n)$ [97, 116]. Intuitivement, cela peut se comprendre de la manière suivante. *A priori*, la fonction $f(n)$ est en $\Omega(n)$, donc le maximum de $f(k)+f(n-k)$ est atteint lorsque k est proche de 0 ou proche de n. Dans la version «boustrophédon», on n'ajoute à ce maximum que le minimum de k et de $n-k$, alors qu'on y ajoute systématiquement k dans la version d'origine, où $f(n) = \max_{0 \leq k \leq n}(f(k) + f(n-k) + k)$.

Les *structures itératives* forment un cas particulier. Sont ainsi qualifiées les structures dont le graphe de dépendance de la spécification combinatoire (non standard) est acyclique. Le graphe est ainsi défini : il existe un arc de l'ensemble A vers l'ensemble B s'ils sont respectivement dans le membre gauche et le membre droit de la même règle. Dans ce cas, la phase de génération est linéaire.

2. Du grec *bous*, bœuf et *strephein*, tourner. Ecriture archaïque (grec, étrusque) qu'on lisait alternativement de gauche à droite et de droite à gauche [2].

	Références	Dénombrement	Génération	Mémoire
Structures décomposables				
Cas général	[86, 87]+[180]	$O(n(\log n)^2 \operatorname{loglog} n)$	$O(n \log n)$	$O(n)$
Cas holonome	[86, 87]+[92]	$O(n)$	$O(n \log n)$	$O(1)$
Cas itératif	[86, 87]+[180]	$O(n(\log n)^2 \operatorname{loglog} n)$	$O(n)$	$O(n)$
Langages algébriques	[92]	$O(n)$	$O(n \log n)$	$O(1)$
	[125]	$O(n^2)$	$O(n)$	$O(n^2)$
Langages rationnels	[105]+[92]	$O(n)$	$O(n)$	$O(1)$

TAB. 1.1 – *Complexités arithmétiques au pire des algorithmes de génération de structures décomposables.*

Théorème 4 (Flajolet, Zimmermann, Van Cutsem 1994 et 1997 [86, 87]) *La génération aléatoire d'une structure décomposable de taille n s'effectue en $O(n \log n)$ opérations arithmétiques dans le pire des cas, après un traitement préliminaire en $O(n^2)$ opérations arithmétiques. La complexité arithmétique en espace est en $O(n)$.*

Si la série génératrice est holonome, alors le traitement préliminaire peut s'effectuer en $O(n)$ opérations arithmétiques.

Si la structure est itérative, alors la phase de génération s'effectue en $O(n)$ opérations.

Une amélioration de l'algorithme dans le cas holonome a été apportée par Goldwurm en 1995 [92]. En fait, son travail concernait uniquement les langages algébriques, mais se généralise immédiatement à tout structure dont la série génératrice est holonome. Goldwurm remarque qu'il n'est pas nécessaire de conserver en mémoire simultanément l'ensemble des coefficients de 1 à n pour chaque ensemble de la spécification. Il suffit d'en connaître le nombre (constant) juste suffisant pour pouvoir calculer les autres avec la récurrence linéaire.

Théorème 5 (Goldwurm 1995 [92]) *Dans le cas holonome, la génération d'un mot de longueur n peut s'effectuer en espace arithmétique $O(1)$, tout en conservant les complexités arithmétiques en $O(n)$ et $O(n \log n)$ respectivement pour les phases de dénombrement et de génération.*

Dans le cas général, la phase de génération peut être très sensiblement améliorée grâce à un progrès récent de l'algorithmique du calcul arithmétique. Joris van der Hoeven a proposé, en 1999, une méthode pour calculer les coefficients jusqu'à l'ordre n de toute série génératrice de structures décomposables en temps $0(M(n) \log n)$, où $M(n)$ désigne la complexité arithmétique de multiplication de deux polynômes de degré $n-1$ [180] [3]. Le meilleur algorithme de multiplication, fondé sur la transformée de Fourier, présente une complexité arithmétique $M(n) = O(n \log n \log \log n)$. Le résultat de van der Hoeven permet donc d'effectuer l'étape de dénombrement en complexité arithmétique $O(n(\log n)^2 \log \log n)$, ce qui est une très nette amélioration par rapport au $O(n^2)$ antérieur.

Autres types de décompositions récursives.

D'autres types de formalismes récursifs généraux ont été développés pour le dénombrement et la génération aléatoire de structures combinatoires. Les *grammaires d'objets* de Dutour et

3. C'est la même fonction M qui désigne la complexité logarithmique de la multiplication de deux nombres en page 4.

Fédou [77] permettent de spécifier et d'engendrer des objets selon plusieurs paramètres simultanément (par exemple nombre de sommets *et* nombre de feuilles pour un arbre). Ces paramètres ne sont pas nécessairement algébriques (c'est-à-dire ne mènent pas systématiquement à des séries génératrices algébriques). L'équipe de Renzo Pinzani a mis au point la méthodologie *ECO*, dans laquelle les objets combinatoires sont représentés par des systèmes de règles qui peuvent être interprétées comme des chemins dans le plan discret [19, 20].

1.2.2 Le cas des langages algébriques.

L'algorithme «historique».

Les langages algébriques, lorsqu'ils ne sont pas intrinsèquement ambigus, constituent des structures décomposables très particulières. Ils sont spécifiés par des grammaires, qui ne font usage que de deux opérateurs : l'union disjointe et le produit de concaténation. Les symboles non terminaux de la grammaire représentent les ensembles de mots de la spécification. Le problème de leur génération aléatoire a été abordé dès 1983 par Hickey et Cohen [105]. Le principe de leur méthodologie était similaire à ce que nous avons vu jusqu'ici. Cependant, comme ils considéraient des grammaires algébriques (non ambiguës) non normalisées, les produits n'étaient pas forcément binaires, contrairement à ceux des spécifications standard. Des convolutions et des boucles multiples apparaissaient donc dans le dénombrement et la génération. Par exemple, il faut calculer $\sum_{k_1+k_2+\ldots+k_l=n} W_1(k_1)W_2(k_2)\ldots W_l(k_l)$ pour compter le nombre $V(n)$ de mots de longueur n dérivant du symbole V dans la règle $V \to W_1W_2\ldots W_l$. Hickey et Cohen présentèrent deux versions de l'algorithme. Dans la première, toutes les probabilités susceptibles d'être utilisées lors de la génération sont calculées lors de la phase de dénombrement ; celle-ci a une complexité arithmétique en $O(n^{r+1})$ (en temps et en espace mémoire), où r désigne le nombre de symboles non terminaux de la grammaire ; la phase de génération se déroule en temps $O(n)$. La seconde version présente une première phase plus légère en $O(n^2 \log n)$, et une complexité de $O((n \log n)^2)$ pour la génération.

Un algorithme de génération linéaire.

Comme les langages algébriques ont des séries génératrices holonomes, l'algorithme général offre des complexités en $O(n)$ pour le pré-traitement et en $O(n \log n)$ pour la génération. Des améliorations ont été apportées par Mairson en 1994 [125] et Goldwurm en 1995 [92]. Nous avons vu que le travail de Goldwurm s'applique en fait à toute structure dont la série génératrice est holonome.

L'équivalent de la spécification standard pour une grammaire algébrique est la forme normale de Chomsky (FNC). Tout membre droit d'une règle est soit le mot vide ε, soit une lettre de l'alphabet terminal, soit un symbole non terminal, soit un produit (concaténation) de deux symboles non terminaux. L'intérêt majeur de cette forme normale réside dans le caractère binaire des produits, qui évite les convolutions multiples.

Le but de Mairson est d'accélérer la phase de génération, au prix d'un traitement préliminaire plus long et plus coûteux en mémoire. L'accélération se place au niveau du choix de k dans les règles $C \to A{\cdot}B$ (page 9). La stratégie consiste à calculer, pour chaque règle $C \to A{\cdot}B$ de la grammaire, tous les nombres $a_k b_{l-k}$ pour $0 \le l \le n$ et $0 \le k \le l$ et certaines de leurs sommes partielles, puis les stocker dans une structure qui peut être parcourue rapidement.

Cette structure est un arbre binaire qui s'inspire du codage de Huffmann.

Théorème 6 (Mairson 1994 [125]) *La génération d'un mot de longueur n s'effectue en $O(n)$ opérations arithmétiques, après un traitement préliminaire en $O(n^2)$ opérations arithmétiques. La complexité arithmétique en espace est en $O(n^2)$.*

Il faut cependant remarquer que, même si la moindre complexité de l'étape de génération est un progrès certain, l'algorithme peut être impraticable dès que la longueur n dépasse quelques centaines de lettres, à cause de la place occupée en mémoire. En effet, une complexité arithmétique en $O(n^2)$ signifie qu'en réalité la mémoire nécessaire est en $O(n^3)$ (voir section 1.2.3).

Hélas, il n'est pas possible de combiner les travaux de Goldwurm et de Mairson pour aboutir à une complexité arithmétique globalement linéaire ! En revanche, les langages rationnels se comportent de façon idéale de ce point de vue :

Théorème 7 (Hickey, Cohen 1983 [105]) *La génération d'un mot de longueur n d'un langage rationnel s'effectue en complexité arithmétique $O(n)$ pour la phase de dénombrement, la phase de génération et la mémoire occupée.*

(En fait, la mémoire occupée peut être en $O(1)$, en vertu du résultat ultérieur de Goldwurm.) Ce résultat peut être vu comme une conséquence du théorème 4, car un langage rationnel ressort à la fois du cas holonome et du cas itératif (une expression rationnelle est une spécification itérative). Notons que la forme normale de Chomsky n'est pas utile en ce qui concerne les langages rationnels. La génération peut s'effectuer plus simplement par construction d'un chemin dans une automate du langage (ce qui revient à employer une grammaire régulière pour engendrer le mot lettre par lettre).

Grammaires ambiguës.

Tous les schémas de génération vus jusqu'à présent imposent que la spécification, ou la grammaire dans le cas algébrique, soit non ambiguë. La difficulté de la génération uniforme à partir d'une grammaire ambiguë est liée au fait que compter les mots de longueur n engendrés par une grammaire ambiguë est un problème #P-complet. Dans [93], on présente un algorithme de génération *presque* uniforme en temps quasi-polynomial. Dans [29], on propose un algorithme de génération exactement uniforme de complexité polynomiale à condition que le degré d'ambiguïté de la grammaire (c'est-à-dire le nombre maximum de dérivations aboutissant à un mot de longueur n) soit polynomial en n. Si le degré d'ambiguïté est borné par une constante, alors la complexité est du même ordre que pour une grammaire non ambiguë. L'algorithme consiste à engendrer un arbre de dérivation (ce qui est possible avec la méthode classique), puis accepter le mot correspondant avec probabilité $1/m$, où m est le nombre d'arbres de dérivation aboutissant au même mot, ou le refuser (et recommencer le processus) avec probabilité $1 - 1/m$.

1.2.3 Une technique paresseuse et fructueuse.

Complexité arithmétique vs complexité logarithmique.

L'approche récursive fait invariablement appel à des calculs de probabilités qui dépendent du nombre d'objets de tailles données. Or ces nombres peuvent augmenter exponentiellement en fonction de la taille. C'est d'ailleurs le cas pour la plupart des structures «intéressantes». Pour certains objets combinatoires, les calculs se simplifient de sorte qu'on ne manipule que des nombres de petite taille, c'est-à-dire dont la longueur en bits est de l'ordre du logarithme de la taille des objets. C'est le cas, par exemple, des arbres binaires [10, 15, 149]. Cependant, en général, le calcul et le stockage de grands nombres est inévitable. Le nombre d'objets de taille n peut occuper une place en mémoire en $O(n \log n)$ dans le cas étiqueté, en $O(n)$ en non étiqueté. Dans ces conditions, seule la mesure de complexité logarithmique peut rendre compte du temps effectif de génération et de l'espace mémoire réellement occupé. Dans ce cadre, la génération aléatoire d'une structure décomposable de taille n s'effectue[4] en temps $O(n^{2+\epsilon})$ dans le pire des cas (si l'algorithme à base de transformation de Fourier est utilisé pour la multiplication), après un traitement préliminaire en temps $O(n^{2+\epsilon})$. La complexité logarithmique en espace est en $O(n^{2+\epsilon})$. Comme précédemment, il existe des structures pour lesquelles ces bornes sont atteintes, et pour lesquelles la complexité moyenne est du même ordre que la complexité au pire. Les simulations effectuées en Maple avec le module Combstruct (anciennement Gaïa [194]) ont montré qu'en pratique, on ne peut pas espérer engendrer en temps raisonnable des objets de plus de quelques centaines d'atomes.

Génération presque uniforme.

Il existe une façon simple et naturelle de résoudre ce problème de vitesse : effectuer les calculs en arithmétique flottante au lieu de manipuler des nombres entiers. Les nombres en virgule flottante, qui peuvent dénombrer des objets de très grande taille si leur format est bien choisi, occupent une place constante en mémoire et les opérations arithmétiques s'effectuent en temps constant. Cependant, cet avantage se paie au prix d'erreurs de calcul dues aux arrondis inévitables lors des opérations. On notait dans [86] que la perte d'uniformité devait être raisonnable car l'algorithme de génération est numériquement stable. D'autre part, Mairson [125] étudiait ce problème dans le cas particulier de son algorithme pour les langages algébriques. Restait à explorer systématiquement cette possibilité pour évaluer à quel point l'uniformité est dégradée. C'est l'un des deux problèmes que Paul Zimmermann et moi-même avons étudiés dans [69].

Rappelons quelques propriétés des calculs en arithmétique flottante [56]. Tout nombre en virgule flottante s'écrit

$$x = s_x \cdot m_x \cdot 2^{e_x},$$

où $s_x \in \{-1,1\}$ est le *signe*, m_x la *mantisse* m_x et e_x l'*exposant*. Comme la mantisse est de taille fixée, les opérations arithmétiques ne donnent pas, en général, des résultats exacts. Cependant, le standard IEEE 754 [1] stipule que toute opération arithmétique de base sur des nombres flottants se comporte comme si on l'effectuait en précision infinie, puis arrondissait le résultat pour pouvoir le représenter en virgule flottante. Le même standard fixe les règles d'arrondi. Les symboles ∇ et Δ désignent respectivement les fonctions d'arrondi «vers $-\infty$» et «vers $+\infty$». Celles-ci satisfont

$$x(1-\varepsilon) \le \nabla(x) \le x \qquad \text{et} \qquad x \le \Delta(x) \le x(1+\varepsilon),$$

où la valeur ε est la *précision machine*. Elle dépend du nombre de bits b de la mantisse : $\varepsilon = 2^{1-b}$.

4. La notation $O(k^{l+\epsilon})$ est équivalente à $O(k^{l+o(1)})$. Par exemple, $k \log k = O(k^{1+\epsilon})$, ainsi que $k \log \log k$ et $k(\log k)^2$.

Notre résultat consiste en une évaluation de la perte d'uniformité due au calcul en nombres flottants. Les considérations de complexité se déduisent immédiatement du théorème 4.

Théorème 8 (Denise, Zimmermann 1999 [69]) *Soit C un ensemble de structures combinatoires dont la spécification standard admet $m+1$ classes. Si l'on utilise une arithmétique de nombres en virgule flottante, alors un objet de taille n est engendré avec une probabilité $\tilde{p}_{m,n}$ telle que*

$$p_n(1-\alpha_{m,n}) \leq \tilde{p}_{m,n} \leq p_n(1-\alpha_{m,n})^{-1}, \quad \textit{avec} \quad \alpha_{m,n} = O(m^2n^32^{-b})$$

où $p_n = 1/c_n$ est la probabilité uniforme sur les éléments de C de taille n et b est la longueur de la mantisse des nombres en virgule flottante. La complexité logarithmique de la phase de dénombrement est en $O(n^2M(b))$; celle de la génération d'un objet est en $O(n \log n M(b))$ dans le pire des cas, où $M(b)$ désigne la complexité au pire des opérations arithmétiques standard sur des nombres en virgule flottante dont la mantisse est de taille b.

Génération exactement uniforme.

Une démarche «paresseuse» permet d'obtenir une génération exactement uniforme à faible coût. Le principe consiste à calculer les coefficients et les probabilités approximativement, en faisant en sorte de connaître un majorant de l'erreur commise. Pour simplifier, prenons un problème de choix binaire, comme dans le cas $C = A + B$ (le propos se généralise aisément aux autres cas). Supposons que l'on ne connaisse pas la probabilité $p_A = a_n/c_n$ de choisir A, mais que l'on sache la calculer approximativement en majorant l'erreur commise. En d'autres termes, on connaît un minorant p_A^- et un majorant p_A^+ de p_A. Pour effectuer le choix, on tire comme dans l'algorithme usuel un nombre aléatoire $U \in [0,1]$. Si $U < p_A^-$ ou $U \geq p_A^+$, alors la décision peut être prise ; sinon, on se trouve dans l'«intervalle d'incertitude». Dans ce dernier cas, deux possibilités sont offertes. On peut calculer une approximation plus fine de p_A et recommencer le processus jusqu'à pouvoir décider si $U < p_A$ ou $U \geq p_A$ (et la complexité au pire n'est pas bornée). On peut aussi se résoudre à calculer exactement la probabilité. Quelle que soit l'option choisie, il est clair que cette approche peut améliorer la complexité moyenne de la génération, pourvu que le calcul de p_A^+ et p_A^- soit assez rapide et que l'intervalle d'incertitude soit suffisamment petit.

J'ai utilisé pour la première fois cette stratégie dans un article publié en 1996 dans lequel je me penchais sur la génération aléatoire de mots de langages rationnels. Je m'intéressais aux langages pour lesquels les séries génératrices correspondant aux états de l'automate fini déterministe minimal ont toutes le même pôle dominant ρ, et ce pôle est simple. Selon le théorème de Perron-Frobenius, il suffit pour cela que l'automate soit fortement connexe et apériodique [85]. Dans ces langages, le procédé de génération lettre par lettre présente la propriété suivante : à tout instant, la probabilité de choisir une lettre a est proche de c/ρ, où c est une constante, et on sait calculer un (petit) intervalle dans lequel cette probabilité se trouve à coup sûr dès que le nombre de lettres à engendrer dépasse une certaine constante. La méthode consiste à calculer une fois pour toutes cet intervalle (pour chaque lettre de l'alphabet et chaque état de l'automate) et à procéder comme décrit dans le paragraphe précédent.

Théorème 9 (Denise 1996 [61] et 2001) *Soit L un langage rationnel dont l'automate fini déterministe minimal est fortement connexe et apériodique. Il existe un algorithme de génération aléatoire des mots de L dont complexité logarithmique moyenne est en $O(P(\log n))$, où*

P est un polynôme, pour le pré-traitement et en $O(n)$ pour la génération. La place mémoire occupée en moyenne est en $O(\log n)$.

Le $O(P(\log n))$ provient du fait qu'il suffit de calculer de l'ordre de $\log n$ bits de c/ρ pour déterminer les intervalles lors de la phase préliminaire. Dans le résultat publié en 1996, ce $O(P(\log n))$ était en fait un $O(P(n))$ (où P, qui dépend de l'automate, peut être de degré arbitraire) ! Avec quelques années de recul, je m'aperçois que $O(\log n)$ bits suffisent là où j'en avais mis $O(n)$. Ceci fait de ce théorème le premier résultat présentant une complexité logarithmique globalement linéaire (en moyenne) pour la génération aléatoire d'une vaste classe de langages.

C'est au cours d'une discussion avec Laurent Alonso et Paul Zimmermann que nous envisageâmes une méthode générale fondée sur des principes similaires pour les structures décomposables. L'idée consistait à calculer les coefficients et probabilités comme dans l'algorithme classique, mais d'abord de façon «paresseuse» : calculer rapidement des approximations en nombres flottants, et n'effectuer des calculs exacts (en nombres entiers) que lorsque c'est absolument nécessaire. Ceci a fait l'objet d'un travail que j'ai effectué en collaboration avec Paul Zimmermann. Le principe de base est le même que ci-dessus. Reprenons l'exemple $C = A + B$. En arithmétique flottante, on ne peut pas calculer la probabilité $p_A = a_n/c_n$ de choisir A. Cependant, en utilisant à bon escient les deux types d'arrondi ∇ et Δ, on sait calculer les bornes p_A^- et p_A^+ de p_A de l'intervalle d'incertitude de p_A. L'option choisie dans le cas où on tombe dans cet intervalle est de recalculer les coefficients en arithmétique exacte. Nous montrons qu'à toute étape de l'algorithme la probabilité de ce recalcul est inférieure à $3(2mn^2 + 3)\epsilon$, donc décroît exponentiellement en fonction de la longueur de la mantisse (notons que pour que cela reste une probabilité, il faut que $3(2mn^2 + 3)\epsilon < 1$). Ceci nous amène au théorème suivant.

Théorème 10 (Denise, Zimmermann 1999 [69]) *La complexité logarithmique moyenne de la phase de dénombrement, en fonction de n et de la longueur b de la mantisse, est*

$$D(n,b) = O(M(n) \log n M(b)).$$

La complexité logarithmique moyenne de la génération d'un objet est

$$G(n,b) = O(n \log n M(b) + n^6 \varepsilon M(n \log n)),$$

où $\varepsilon = 2^{1-b}$ et $M(k)$ désigne le coût de multiplication de deux nombres de k bits. La place mémoire utilisée en moyenne est en $O(nb + n^6 \varepsilon \log n)$. Les complexités au pire en temps et en espace sont identiques à celles de l'algorithme en arithmétique exacte.

Dans l'article d'origine, la complexité du dénombrement était en $O(n^2 M(b))$. Ce sont les techniques de manipulation de séries génératrices de Joris van der Hoeven (page 10) qui permettent d'aboutir à $D(n,b) = O(M(n) \log n M(b))$.

Le résultat de ce théorème s'avère particulièrement intéressant si la longueur de la mantisse peut être modifiée en fonction de la taille des structures à engendrer :

Corollaire 11 (Denise, Zimmermann 1999 [69]) *Si $\varepsilon = 1/n^7$, alors*

$$G(n,b) = O(n \log n M(\log n)).$$

D'un point de vue pratique, les expérimentations ([69, tables 1 à 4]) ont montré qu'on gagne facilement un facteur n^2 dans le temps de génération. Grâce à la participation de Sylvie Corteel, Isabelle Dutour et Frédéric Sarron, un module MuPAD de génération nommé CS a vu le jour (et est encore en développement) [48, 64]. Par rapport au module Combstruct de Maple, CS ajoute la possibilité du calcul en arithmétique flottante, et la faculté d'engendrer automatiquement des programmes C de génération aléatoire. Le module CS est actuellement utilisé dans le cadre de recherches en test statistique de logiciels que je présenterai en section 1.5.

La partie théorique de notre travail a montré que les calculs en arithmétique flottante ont un véritable intérêt dans le domaine de la génération aléatoire. Nous avons confirmé et quantifié le fait que, dans le cas où on ne cherche pas une uniformité rigoureuse, on peut sans problème écrire des programmes «légers» dans des langages standard. D'autre part, l'uniformité exacte peut être atteinte avec un fort gain en temps par rapport aux algorithmes usuels, en combinant arithmétiques flottante et exacte. Cependant, notre méthode n'améliore que la complexité moyenne.

D'un point de vue pratique, il n'existe pas encore de programme qui adapte la longueur de la mantisse à la taille des données, comme suggéré par le Corollaire 11. Il est maintenant possible de le réaliser, grâce à la bibliothèque MPFR [55]. Celle-ci permet de manipuler des nombres en arithmétique flottante avec une précision fixée par l'utilisateur.

1.2.4 Génération non uniforme contrôlée.

Une extension naturelle de la génération aléatoire uniforme est la génération... non uniforme ! Il s'agit d'engendrer des objets combinatoires selon une loi de probabilité donnée. Jusqu'à récemment, ce problème n'avait jamais été étudié [5]. C'est dans le cadre de la bio-informatique des génomes que je m'y suis intéressé, en collaboration avec Olivier Roques et Michel Termier [66]. Comme nous le verrons dans le chapitre 2, les séquences aléatoires occupent une place privilégiée en génomique.

Le problème est le suivant : il s'agit de tirer au hasard des mots d'un langage de façon à respecter une distribution donnée sur les nombres d'occurrences de chacune des lettres de l'alphabet. Plus précisément, on se donne un langage L sur un alphabet $X = \{x_1, x_2, \ldots, x_k\}$ et un entier n. Notons L_n l'ensemble des mots de L de longueur n. Le problème consiste à engendrer des mots de L_n de façon à respecter une distribution des lettres donnée par un vecteur de k nombres positifs. Deux variantes peuvent être considérées :

1. *Génération en fréquences moyennes.* Les mots engendrés doivent respecter *en moyenne* une distribution donnée par un vecteur $\mathbf{v} = (v_1, \ldots, v_k)$ tel que $v_1 + \ldots + v_k = 1$. Plus précisément, il s'agit de tirer aléatoirement des mots de L_n de façon que
   (a) tout mot de L_n ait une probabilité non nulle d'être engendré ;
   (b) pour tout $i \in \{1, 2, \ldots, k\}$, la proportion de lettres x_i d'un mot engendré soit égale (ou asymptotiquement équivalente) à v_i : si $p(w)$ est la probabilité que le mot w soit engendré par l'algorithme, on doit avoir

$$\frac{1}{n} \sum_{w \in L_n} |w|_{x_i} p(w) \;\sim\; v_i \quad \forall i \in \{1, 2, \ldots, k\}.$$

5. sauf pour ce qui est de la génération aléatoire de nombres, sur laquelle il existe une abondante littérature [70].

   (c) deux mots ayant la même distribution des lettres aient la même probabilité d'être engendrés.

2. *Génération en fréquences exactes.* La distribution du nombre de lettres de tout mot engendré doit respecter exactement un vecteur d'entiers $(n_1,\dots,n_k)$ donné. Il s'agit d'un tirage aléatoire uniforme dans un sous ensemble de L_n constitué de tous les mots $w \in L_n$ tels que $|w|_{x_i} = n_i$ pour tout $i \in \{1,2,\dots,k\}$.

On peut aussi alléger les contraintes en fixant uniquement les fréquences de certaines lettres et laissant les autres «libres». C'est en vue de la génération de séquences génomiques que nous nous sommes focalisés sur les langages. Toutefois le problème se généralise aux structures décomposables générales, en considérant une distribution sur les atomes.

Notre principale contribution concerne la génération en fréquences moyennes. Notre démarche consiste à munir chaque lettre de l'alphabet d'un *poids* en définissant une application $\pi : X \to \mathbb{R}_+^*$, que nous appelons *pondération*. Elle s'étend naturellement à tous les mots de X^* ainsi qu'à tout langage fini, en particulier à L_n :

$$\pi(w) = \prod_{1\le i\le k} \pi(x_i)^{|w|_{x_i}} \qquad \text{et} \qquad \pi(L_n) = \sum_{w\in L_n} \pi(w).$$

Il est clair que si l'algorithme de génération est tel que

$$p(w) = \frac{\pi(w)}{\pi(L_n)} \quad \forall w \in L_n,$$

alors le nombre moyen d'occurrences d'une lettre donnée dans les mots engendrés est d'autant plus grand que la pondération de cette lettre est élevée par rapport aux autres. Il est clair aussi que les conditions (1a) et (1c) ci-dessus sont respectées.

Lorsque l'on dispose d'une pondération π adéquate, la démarche pour engendrer des mots aléatoires est identique à celle utilisée pour la génération uniforme, à ceci près que, lors de la phase de dénombrement, le poids des lettres est pris en compte. On dit que la grammaire est *pondérée* par π. Pour un symbole non terminal T, Le coefficient $T(n)$ ne compte pas le nombre de mots de taille n, mais la somme de leurs poids. Il suffit pour cela de modifier très légèrement les règles de récurrence classiques. L'étape de génération est identique au cas uniforme. On garantit de cette façon que la probabilité d'apparition de chaque mot sera proportionnelle à son poids $\pi(w)$.

Le problème est de trouver une pondération qui satisfait la condition (1b). Considérons la série génératrice

$$L_\pi(t,\mathbf{x}) = \sum_{w\in L} \pi(w) t^{|w|} x_1^{|w|_{x_1}} \dots x_k^{|w|_{x_k}},$$

où $\mathbf{x} = (x_1,x_2,\dots,x_k)$. Pour toute lettre $i \in \{1,\dots,k\}$, le nombre moyen d'occurrences de la lettre x_i dans les mots engendrés par l'algorithme est égal à $n\mu_i(\pi)$, où

$$\mu_i(\pi) = \frac{\sum_{w\in L_n} |w|_{x_i} \pi(w)}{n\pi(L_n)} = \frac{[t^n]\Gamma_{\pi,x_i}(t)}{[t^n]\Gamma_\pi(t)}$$

avec

$$\Gamma_{\pi,x_i}(t) = \frac{\partial L_\pi(t,\mathbf{x})}{\partial x_i}(t,\mathbf{1}) \qquad \text{et} \qquad \Gamma_\pi(t) = t\frac{\partial L_\pi(t,\mathbf{x})}{\partial t}(t,\mathbf{1})$$

Il faut donc trouver une pondération π telle que $\mu_i(\pi) \sim v_i$ pour tout $i \in \{1,\dots,k\}$. Si L est algébrique non rationnel et sa si grammaire est *fortement connexe* (c'est-à-dire si tout symbole non terminal peut être obtenu par dérivations à partir de tout autre symbole non terminal), alors un résultat de Drmota [73] permet de réduire le problème à la résolution de systèmes d'équations algébriques [66, Théorème 1]. Cependant, on ne peut les résoudre en général. En revanche, nous avons pu identifier une classe intéressante de langages rationnels pour lesquels une résolution numérique est toujours possible.

Théorème 12 (Denise, Roques, Termier 2000 [66]) *Soit L un langage rationnel sur un alphabet $X = \{x_1, \cdots, x_k\}$ et $\mathcal{A}$ son automate fini déterministe minimal. Si chaque composante fortement connexe maximale non triviale de $\mathcal{A}$ est apériodique et contient toutes les lettres de X, alors les $\pi(x_i)$ qui respectent les conditions (1a), (1b) et (1c) sont solutions d'un système d'équations algébriques à coefficients réels.*

Ce résultat est lié, encore une fois, au théorème de Perron-Frobenius : les séries génératrices des langages rationnels en question ont un unique pôle de module minimal (éventuellement multiple). C'est ce qui permet de donner une expression asymptotique de $\mu_i(\pi)$ en fonction de l'inconnue π_i.

Ce travail est un tout premier pas dans la génération non uniforme contrôlée d'objets combinatoires. Il est possible que le cas des langages algébriques (ou au moins de langages plus généraux que dans le théorème ci-dessus) puisse être résolu par une voie analytique ou algorithmique différente de celle que nous avons tentée. Du côté applicatif, l'algorithme de génération non uniforme est implanté dans le logiciel de génération de séquences génomiques aléatoires «structurées» décrit en section 2.5.4.

1.3 Méthodes à rejet.

1.3.1 Rejet classique.

Soit E l'ensemble dans lequel nous voulons engendrer des structures combinatoires. Supposons que nous connaissons un ensemble F qui inclut E et dans lequel nous savons faire de la génération aléatoire. Alors, pour engendrer un objet aléatoire dans E, il suffit de tirer une suite d'objets dans F jusqu'à ce qu'on en trouve un qui appartient à E. Cette méthode peut être naturellement envisagée lorsqu'on ne sait pas engendrer directement des objets de E. De plus, même pour des structures décomposables, elle constitue parfois une alternative plus efficace que l'approche récursive.

Si le tirage est uniforme dans F, alors il est uniforme dans E. Le nombre moyen d'essais pour aboutir à un élément de E est $|F|/|E|$. La complexité dans le pire des cas n'est pas bornée. C'est pourquoi on ne parle pas d'algorithme dans ce cadre, mais de pseudo-algorithme [6].

L'exemple canonique d'utilisation de cette approche est probablement celui des graphes étiquetés connexes [70, 173]. On sait depuis les travaux d'Erdös et Renyi que la probabilité qu'un graphe aléatoire (tiré uniformément) à n sommets soit connexe tend vers 1 lorsque $n \to \infty$ [31]. Donc, pour engendrer un graphe connexe, commençons par tirer aléatoirement un graphe quelconque en passant en revue chaque couple de sommets et décidant s'ils seront reliés par une arête avec probabilité 1/2. Asymptotiquement, la complexité logarithmique moyenne de la méthode est $n(n-1)/2$. En pratique, l'algorithme est efficace même pour un n petit.

6. Toutefois, on remarque dans [6, p.132] que, si on connaît un *vrai* algorithme de génération des objets de E, la complexité au pire de la méthode de rejet peut être bornée par celle de l'algorithme : il suffit de décider qu'à partir d'un certain nombre d'essais avortés, on abandonne la technique de rejet pour l'algorithme.

Ce type d'approche a été appliqué avec succès par Wormald pour engendrer des graphes non étiquetés [189] (il s'inspire pour ce faire d'une méthode due à Dixon et Wilf [71] et l'améliore). Les polyominos convexes sont codés pas des mots d'un langage algébrique [57] ; ils peuvent donc être engendrés par un algorithme récursif classique. Pour le réseau carré, Hochstattler, Loebl et Moll ont conçu un pseudo-algorithme plus efficace : sa complexité logarithmique est linéaire [106]. En 1997, Christoph Dürr, Fouad Ibn-Majdoub-Hassani et moi-même avons effectué le même type de travail sur différentes classes de polyominos convexes en réseau hexagonal [63]. Laurent Alonso utilise le rejet pour tirer des chemins de Motzkin[7] en temps linéaire [5]. Sa méthode est particulière car le rejet s'applique non pas à la génération du chemin, mais à celle du nombre de pas horizontaux qu'il contiendra. Penaud et Roques [138] donnent un algorithme de complexité linéaire pour la génération des mots du langage de Fibonacci : $(a+bb)^*$. Ce langage entre dans le cadre du théorème 9, donc il existe un algorithme récursif en temps linéaire. Cependant, la méthode de Penaud et Roques a le mérite d'être plus simple à implémenter. De plus, elle est inédite car ils tirent astucieusement parti du fait que, pour engendrer un élément uniformément dans E, il n'est pas nécessaire que le tirage soit uniforme dans le sur-ensemble F ; il suffit que tous les éléments de E aient la même probabilité d'apparaître.

1.3.2 Rejet anticipé.

En 1991, Barcucci, Pinzani et Sprugnoli présentèrent une méthode à rejet particulière pour la génération des chemins facteurs gauches de Motzkin [21, 23][8]. Cette approche fut plus tard baptisée «méthode florentine». L'idée, très simple, est la suivante : à chaque étape, tirer une lettre x, $\bar{x}$ ou a avec probabilité $1/3$; ne pas attendre d'avoir engendré n lettres pour rejeter éventuellement le mot, mais le rejeter dès qu'il est manifeste qu'on n'aboutira pas à un facteur gauche de Motzkin, c'est-à-dire dès que la hauteur du chemin correspondant devient négative. En moyenne et asymptotiquement, $2n$ lettres sont tirées, alors que la méthode de rejet classique présente une complexité moyenne de $\sqrt{\pi/3}\, n^{3/2}$. Les facteurs gauches de Motzkin codent pour les animaux dirigés, structures combinatoires qui apparaissent dans des modèles de physique statistique [30, 183]. Le gain en complexité est crucial car le pseudo-algorithme offre la possibilité d'en engendrer de très grands pour en mesurer expérimentalement des caractéristiques.

A l'époque, ce résultat étonnait par sa simplicité. Il semblait très lié à des propriétés combinatoires spécifiques des facteurs gauches de Motzkin, mais cela n'apparaissait pas clairement dans la preuve. Mon directeur de thèse, Jean-Guy Penaud, me chargea de rechercher ces propriétés et de déterminer s'il pouvait y avoir matière à généralisation. Bien sûr, l'idée de base peut être aisément généralisée à toute méthode à rejet[9] (on la verra appliquée plus bas à la génération de cartes planaires). Cependant, dans la plupart des cas, la complexité moyenne de la méthode florentine ne peut être évaluée exactement. J'ai déterminé une classe de langages pour laquelle ce calcul peut être effectué, ainsi que celui de la variance.

7. Un chemin de Motzkin est identique à un chemin de Dyck, à ceci près qu'un troisième pas y est autorisé : le pas horizontal (+1,0). A tout chemin on peut associer un mot en remplaçant respectivement les pas (+1,+1), (+1, − 1) et (+1,0) par les lettres x, $\bar{x}$ et a.

8. Un facteur gauche de Motzkin est un chemin de Motzkin tronqué.

9. mais il est souvent plus coûteux de vérifier à chaque étape que l'objet en cours de génération reste dans l'ensemble adéquat que de le déterminer une fois pour toutes en fin de traitement.

Théorème 13 (Denise 1996 [60]) *Soient A un alphabet de cardinal k, et L un langage sur A qui vérifie : (i) pour tout mot $u \in L$, il existe un mot $v \in L$ tel que u est un facteur gauche* propre *de v (i.e $u \neq v$ et u est facteur gauche de v) ; (ii) tout facteur gauche d'un mot de L est dans L. Alors le nombre moyen de lettres tirées par la méthode florentine pour engendrer un mot de longueur n est*

$$C(n) = \frac{[t^n]\left(\frac{t}{1-t}\mathcal{L}(\frac{t}{k})\right)}{[t^n]\mathcal{L}(\frac{t}{k})}.$$

où $\mathcal{L}(t)$ est la série génératrice de L.

Le mérite de ce résultat est de préciser clairement quelles sont les propriétés combinatoires qui rendent possible le calcul de la complexité. Outre le langage de Motzkin, il existe d'autres langages intéressants qui respectent les propriétés ci-dessus. C'est le cas des chemins sous-diagonaux [22], généralisation des mots de Motzkin, ainsi que des langages de Yamanouchi [96] qui codent les tableaux de Young, structures combinatoires particulièrement riches (cependant la méthode florentine n'est pas efficace en ce qui concerne ces derniers).

Voici une application du rejet anticipé à des objets qui ne relèvent pas du théorème ci-dessus. Une *carte planaire* est une représentation d'un multigraphe planaire sur le plan sans croisement d'arêtes, aux déformations continues du plan près. Les cartes étudiées en combinatoire sont généralement *enracinées* : un sommet et une arête qui lui est adjacente sont distingués. Le nombre de cartes planaires à n arêtes est donné par une formule très simple :

$$p_n = \frac{2}{n+2}\,\frac{3^n}{2n+1}\binom{2n+1}{n}.$$

Il existe plusieurs preuves de cette formule. La première est due à Tutte [178]. Celle de Cori et Vauquelin [47] est bijective : ils établissent une correspondance entre les cartes et une famille d'arbres appelés *arbres bien étiquetés*, et un codage de ces arbres par des mots d'un langage L (non algébrique). Puis ils montrent que la formule ci-dessus compte les mots de longueur $2n$ de L. En 1993, j'ai conçu un pseudo-algorithme de complexité moyenne $O(n^2)$ ($O(n\sqrt{n})$ conjecturé) pour engendrer des cartes planaires [58, 59]. Il consistait à engendrer des mots de L par une méthode de rejet anticipé ; puis un algorithme de décodage donnait la carte correspondante en temps linéaire. Ce fut la première méthode efficace pour engendrer ce type d'objets. En 1998, Gilles Schaeffer a revisité ce problème et amélioré sa solution en concevant de beaux algorithmes de complexité linéaire [157, 158]. Sa méthode s'applique à plusieurs autres familles de cartes (cartes biparties, cartes non séparables, ...) qui sont comptées aussi par des formules multiplicatives simples. Elle est fondée sur un codage des cartes par des arbres particuliers appelés *arbres équilibrés.*

1.3.3 Extraction/rejet et modèle de Boltzmann.

D'autres familles de cartes sont comptées par des formules plus complexes. C'est le cas, par exemple, des cartes sans boucle (c'est-à-dire sans arête d'un sommet à lui-même). On peut imaginer de les engendrer par rejet à partir des cartes générales. Hélas, cela débouche sur une complexité exponentielle. Gilles Schaeffer a imaginé, dans ce cas, la surprenante méthode d'extraction-rejet [157, 158]. Pour engendrer une carte sans boucle à n arêtes, on engendre une carte ordinaire à $m > n$ arêtes ; puis on supprime les boucles (étape d'extraction) ; si la carte obtenue n'a pas n arêtes, on la rejette et réitère le processus. Aussi incroyable que cela paraisse, ce pseudo-algorithme a une bonne complexité moyenne : elle est en $O(n^{5/3})$ pourvu que m soit bien choisi. Dans le cas présent, il faut que $m = 3n/2$. Ce schéma se généralise à plusieurs familles de cartes (non séparables, simples, 3-connexes...), avec le même ordre de complexité. Dans chaque cas, la valeur optimale de m est déterminée analytiquement.

Très récemment, Duchon, Flajolet, Louchard et Schaeffer ont posé les bases d'une nouvelle méthode générale pour la génération de structures décomposables, fondée en partie sur le rejet [74]. Elle est liée aussi au principe d'extraction/rejet, du fait que la taille des objets engendrés n'est pas fixée *a priori*. Tout objet de l'ensemble, quelle que doit sa taille, a une probabilité non nulle d'être engendré. Un paramètre réel x permet de faire varier la loi de probabilité pour fixer la taille moyenne des structures engendrées. Enfin, une étape de rejet permet de sélectionner les objets de taille adéquate. Les auteurs ont donné à ce nouveau schéma de génération le nom de “modèle de Boltzmann” car les probabilités associées aux objets engendrés s'écrivent de façon similaire à celles des configurations d'énergie en physique statistique, selon les travaux de Boltzmann.

Ces deux méthodes (extraction/rejet et modèle de Boltzmann) peuvent être extrêmement efficaces, surtout si l'on accepte d'engendrer des objets dont la taille n'est pas exactement n, mais est comprise dans un intervalle donné. Par exemple, la méthode d'extraction/rejet permet d'engendrer en temps moyen linéaire des cartes de taille $n \pm n^{2/3}$. Cependant, deux objets de tailles différentes ont des probabilités différentes d'apparaître (mais la génération reste uniforme pour les objets de même taille).

1.4 Chaînes de Markov.

1.4.1 Génération presque uniforme.

Les objets combinatoires que nous avons vus jusqu'ici ont ceci de commun qu'on sait les *compter*. Plus exactement, on peut déterminer en temps polynomial le nombre d'objets de taille n, au moins asymptotiquement. Jerrum, Valiant et Vazirani [112] ont montré que, dans ce cas, il est toujours possible de concevoir un générateur aléatoire uniforme de complexité polynomiale[10]. Il existe une foule de structures combinatoires que l'on ne sait pas compter aussi facilement. Pour certaines, le problème de leur dénombrement est $\#P$-complet (la $\#P$-complétude est au dénombrement ce que la NP-complétude est à la décision).

Pour ces cas difficiles, la génération aléatoire *presque* uniforme peut être envisagée. Elle est d'autant plus motivée que, sous certaines conditions, un algorithme de génération presque uniforme peut mener à un algorithme probabiliste polynomial de dénombrement approximatif [112]. Pour un ensemble E d'objets combinatoires, il s'agit de faire en sorte que la différence relative entre la probabilité $p(e)$ d'engendrer un objet $e \in E$ et la probabilité uniforme $p_u = 1/|E|$ soit inférieure à un réel ε fixé :

$$\max_{e \in E} \frac{|p(e) - p_u|}{p_u} \leq \varepsilon.$$

(Bien entendu, $|E|$ est inconnu.) La méthodologie employée est la suivante. On considère une chaîne de Markov dont les éléments de E sont les états, et dont chaque transition est déterminée par une modification d'un objet. Par exemple, si les objets sont des graphes,

10. ce qui ne préjuge pas, bien sûr, du degré du polynôme.

chaque transition peut consister à ajouter ou retirer une arête avec une certaine probabilité. Si la chaîne est irréductible (de tout état on peut joindre tout autre état) et apériodique, alors la loi du processus tend vers une unique distribution stationnaire, quelle que soit la distribution au départ. De plus, si la probabilité de transition de e à e' est égale à celle de la transition inverse pour tous $e,e' \in E$, alors la distribution stationnaire est la distribution uniforme. Donc, l'algorithme de génération consiste à partir d'un état quelconque et suivre les transitions avec les probabilités correspondantes «pendant un certain temps». Le problème est de déterminer le temps suffisant pour que la génération soit uniforme à ε prés. S'il est polynomial en la taille n des objets à engendrer et en $\log \frac{1}{\varepsilon}$, on dit que la chaîne *se mélange rapidement*. Prouver qu'une chaîne de Markov se mélange rapidement est loin d'être facile. Le premier résultat positif est dû à Jerrum et Sinclair [110, 165], qui ont présenté en 1989 un algorithme polynomial pour la génération presque uniforme de couplages parfaits dans un graphe. Ce travail avait pour but de donner un algorithme de calcul approché du permanent d'une matrice 0-1, qui est égal au nombre de couplages parfaits du graphe représenté par la matrice. Ces méthodes ont donné lieu à d'autres développements et applications, dont on peut trouver une synthèse dans [111]. Toutefois, les algorithmes pour lesquels on sait calculer le temps de mélange sont rares. Il existe toutefois des méthodes heuristiques pour déterminer un temps à partir duquel la chaîne a de bonnes chances d'être mélangée [190].

1.4.2 Graphes planaires aléatoires.

Les graphes planaires sont des objets bien plus difficiles à appréhender que les cartes planaires. Aucune formule de dénombrement (même asymptotique) n'est connue. En 1996, Marcio Vasconcellos, Dominic Welsh et moi-même avons fait un premier pas dans l'étude des propriétés de graphes planaires aléatoires [68]. Nous avons commencé par concevoir une chaîne de Markov pour leur génération, dans le cas étiqueté. Son principe est très simple : partir d'un graphe planaire quelconque à n sommets. A chaque étape, choisir au hasard une paire de sommets. Si c'est une arête, l'ôter du graphe. Sinon, l'ajouter au graphe à condition que celui-ci reste planaire. Nous n'avons pas pu démontrer que cette chaîne de Markov se mélange rapidement. Cependant, les simulations que nous avons effectuées nous ont amenés à conjecturer certains résultats concernant les propriétés asymptotiques du graphe planaire aléatoire à n sommets, que nous noterons R_n. Les résultats expérimentaux suggèrent que

- le nombre moyen d'arêtes de R_n, noté $e(R_n)$, satisfait : $e(R_n) \sim Cn$ avec $C \approx 2$;
- la probabilité que R_n soit connexe tend vers 1 lorsque n tend vers l'infini ;
- la probabilité que R_n soit 2-connexe tend vers 0 lorsque n tend vers l'infini.

Notre principal résultat à ce propos concerne le nombre d'arêtes.

Théorème 14 (Denise, Vasconcellos, Welsh 1996 [68])

$$\frac{3n-6}{2} \leq e(R_n) \leq 3n-6.$$

La seconde inégalité est triviale : $3n-6$ est le nombre d'arêtes d'un graphe planaire connexe maximal. La première se démontre en examinant la structure de l'ensemble des graphes planaires partiellement ordonné par la relation «est sous-graphe de». Nous avons aussi obtenu le premier résultat de dénombrement des graphes planaires. Dans le théorème suivant, nous nous plaçons dans le cas non étiqueté.

Théorème 15 (Denise, Vasconcellos, Welsh 1996 [68]) *Il existe un réel θ tel que le nombre $u_c(n)$ de graphes planaires connexes non étiquetés à n sommets satisfait :*

$$\lim_{n\to\infty} (u_c(n))^{\frac{1}{n}} = \theta, \qquad et \qquad \frac{256}{27} \leq \theta \leq 8\frac{256}{27}.$$

1.4.3 Génération exactement uniforme.

Dans certains cas, l'approche par chaîne de Markov aboutit à une distribution exactement uniforme. Le problème de la génération d'arbres couvrants d'un graphe en est une belle illustration. De plus, il a des applications en génomique, comme nous le verrons en section 2.5.2. Un arbre couvrant d'un graphe est sous-graphe connexe sans cycle qui contient tous les sommets du graphe. On sait compter les arbres couvrants d'un graphe, et plusieurs algorithmes de génération en temps polynomial sans chaînes de Markov ont été conçus [46, 101]. Toutefois, l'algorithme présenté ci-dessous a le mérite d'être à la fois extrêmement simple et plus performant que les précédents. Il a été découvert indépendamment par Aldous [4] et Broder [39]. Le procédé consiste à construire l'arbre arête par arête comme suit : partir d'un sommet quelconque du graphe ; à chaque étape, choisir uniformément une arête adjacente au sommet courant et la traverser ; si elle n'appartient pas déjà à l'arbre et si elle n'y occasionne pas de circuit, alors l'ajouter à l'arbre ; stopper dès que l'arbre couvre tous les sommets. Le processus peut être vu comme une chaîne de Markov dont les états sont les sous-arbres du graphe enracinés au sommet courant. En utilisant cette représentation, on montre qu'on obtient un arbre couvrant avec probabilité uniforme en complexité moyenne égale au «temps de couverture» du graphe (le temps mis pour parcourir au moins une fois chaque sommet), qui est en $O(n \log n)$ pour presque tous les graphes, et $O(n^3)$ pour les pires d'entre eux [39]. En 1996, Wilson a conçu une variante plus performante [187]. Propp et Wilson donnent dans [140, 141] un cadre assez général dans lequel la génération exactement uniforme avec des chaînes de Markov peut être effectuée. Le serveur de Wilson [186] contient une bibliographie commentée du sujet.

1.5 Une application au test statistique de programmes.

Je décris ici un travail que j'effectue en collaboration avec Marie-Claude Gaudel, Sandrine-Dominique Gouraud et Bruno Marre, de l'équipe *Programmation et Génie Logiciel* du LRI. Le travail a commencé dans le cadre du stage de DEA de Sandrine-Dominique Gouraud [94]. Notre méthodologie et ses premiers résultats sont présentés dans [95]. Le test de logiciel est une activité absolument nécessaire dans toute démarche de développement informatique. Toutes les méthodologies de gestion de projet de développement logiciel comprennent plusieurs étapes de test à différents stades d'avancement du travail. Schématiquement, tester un programme (ou une procédure) consiste à l'exécuter sur des données «bien choisies» puis à confronter les résultats obtenus à ceux attendus. Une part des travaux en génie logiciel a pour but de rationaliser les méthodes de test pour les rendre les plus efficaces possibles. En particulier, le problème de l'automatisation du choix des données fait l'objet d'actives recherches. C'est dans ce cadre que se situe notre travail. L'utilisation de la génération aléatoire d'objets combinatoires constitue une approche totalement nouvelle du problème. Par rapport aux

```
Algorithme Fusion
input: t1, t2: array of int; IMax, JMax:int
output: t3 array of int
vars: i, j, k: int
begin
 i:=1; j:=1; k:=1;
 while((i≤IMax) and (j≤JMax)){
  if(t1[i]<t2[j])
   then{t3[k]:=t1[i];i:=i+1}
   else{t3[k]:=t2[j];j:=j+1};
  k:=k+1};
 while(i≤IMax){
  t3[k]:=t1[i];
  i:=i+1; k:=k+1};
 while(j≤JMax){
  t3[k]:=t2[j];
  j:=j+1; k:=k+1};
end
```

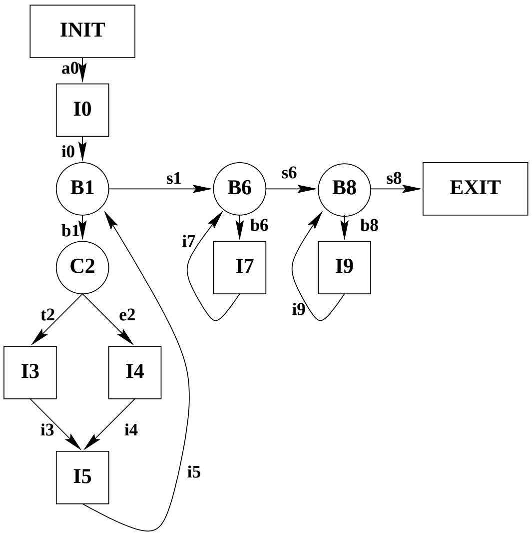


FIG. 1.3 – *Un algorithme et son graphe de contrôle*

méthodes connues, elle mène à une automatisation plus poussée de la génération des jeux de test. Il existe trois principaux types de test :

- le *test fonctionnel* (ou «boîte noire») : on se base uniquement sur les spécifications du programme (ou de la procédure) pour engendrer des données d'entrée.
- le *test structurel* (ou «boîte de verre») : on se base sur la structure du programme (ou de la procédure). Les données sont choisies de façon à respecter un *critère de couverture* de cette structure, préalablement défini.
- le *test statistique* : on se base sur une distribution de probabilités sur le domaine d'entrée du programme.

On peut combiner le test statistique avec l'une des deux premières approches : la distribution est définie en prenant en compte soit les spécifications, soit la structure du programme. On parle alors de test statistique fonctionnel ou de test statistique structurel. Nous nous intéressons précisément au test statistique structurel. La structure sur laquelle nous nous basons est le graphe de contrôle du programme (figure 1.3). Les critères de couverture déterminent les conditions que doit respecter l'ensemble des données avec lesquelles le programme sera testé. Les critères les plus courants sont :

- Tous les chemins : tous les chemins d'exécution (c'est-à-dire les chemins correspondant à une exécution du programme) possibles du graphe doivent être testés.
- Toutes les instructions : chaque instruction doit être exécutée au moins une fois (cela revient à passer au moins une fois par chaque sommet du graphe).
- Tous les enchaînements : chaque enchaînement d'instructions doit être parcouru au moins une fois (cela revient à passer au moins une fois par chaque arc du graphe).

Concevoir un jeu de tests consiste à construire un ensemble de données d'entrée du programme $D = \{d_1,d_2,\ldots,d_N\}$ sur lesquelles le programme va être successivement exécuté. L'entier N est une donnée du problème : c'est le nombre maximal de tests que l'on peut faire. En pratique, on fixe une longueur maximale n des chemins d'exécution, ou un nombre maximum de passages par les boucles. C'est indispensable dans le critère «tous les chemins» pour que les nombre de chemins à tester soit fini. Dans les deux autres critères, on ajoute la condition suivante : tout chemin d'exécution doit avoir une probabilité non nulle d'être parcouru par une exécution. Pour respecter le critère «tous les chemins», il est nécessaire d'engendrer tous les chemins de taille inférieure à n. Ce n'est envisageable en pratique que dans des cas très particuliers. C'est pourquoi nous nous concentrons sur les deux autres critères. Notre méthode est très schématiquement décrite par l'algorithme ci-dessous, dans lequel un «élément» représente une instruction ou bien un enchaînement, selon le critère choisi.

```
i ← 1
tant que i ≤ N faire
début
    tirer au hasard un élément e
    engendrer uniformément au hasard un chemin c passant par e
    si on peut trouver une donnée d'entrée d_i correspondant au chemin c, alors
    début
        écrire d_i
        i ← i + 1
    fin
fin
```

Les chemins considérés sont ceux de longueur inférieure ou égale à n qui vont du sommet initial (INIT) au sommet terminal (EXIT). Il convient de préciser ce que nous entendons par «tirer au hasard un élément e». Le but est d'effectuer les tirages de façon à ce que tous les éléments soient testés aussi équitablement que possible : il faut éviter que certains éléments soient insuffisamment visités par rapport aux autres. En d'autres termes, les probabilités de choisir les différents éléments doivent être les plus équilibrées possibles. Formellement, cela s'exprime en terme de *qualité de test*, notion définie dans [171]. Soit E l'ensemble des éléments et, pour $e \in E$, notons $p(e)$ la probabilité de parcourir e au cours d'un test. La *qualité de test*, notée q_N, dépend des $p(e)$ et du nombre de tests effectués, N. Elle est ainsi définie :

$$q_N = 1 - (1 - \min_{e \in E} p(e))^N.$$

Pour améliorer q_N, il faut augmenter $\min_{e \in E} p(e)$. A cette fin, on peut influer sur les lois du tirage aléatoire des éléments. On remarque que, dans un graphe de contrôle, certains d'entre eux sont parcourus par tous les chemins. C'est le cas des sommets I0, B1, B6 et B8 dans notre exemple. D'autres, comme C2 et I5 sont liés, dans le sens où si l'un est parcouru, l'autre l'est aussi. Ces considérations impliquent que l'on peut réduire l'ensemble des éléments à tirer tout en respectant le critère de couverture, et en améliorant du même coup la qualité de test. On montre aisément que, dans le critère «toutes les instructions», si on tire uniformément les éléments dans $S = \{I0, I3, I4, I7, I9\}$, alors la qualité de test est optimale (et tout chemin d'exécution a une probabilité non nulle d'être exécuté, comme requis). Sandrine Gouraud a développé une heuristique qui, étant donné un graphe de contrôle, détermine l'ensemble S. Elle semble optimale dans la plupart des cas, mais certains graphes «pathologiques» résistent au traitement.

Le graphe de contrôle peut être vu comme un automate fini déterministe. De ce fait, les probabilités $p(e)$, nécessaires pour évaluer la qualité de test, peuvent être aisément calculées par des séries génératrices rationnelles ou des récurrences. De même, la génération d'un chemin se ramène à une génération d'un mot dans un langage rationnel. Une transformation simple du graphe permet de réduire le problème de génération de chemins de longueur inférieure ou égale à n à celui, classique, de génération de chemins de longueur exactement égale à n. Le graphe des chemins passant par un élément donné se détermine par intersection de langages. Concrètement, la spécification des graphes de contrôle et la génération des chemins sont effectuées à l'aide du module CS (page 16).

L'étape de construction des données correspondant à un chemin est particulièrement délicate. Le chemin est d'abord traduit en prédicats. Puis on applique une méthode de résolution de contraintes. En l'occurrence, c'est une adaptation de l'outil GATeL [129] qui est utilisée. Dans la plupart des cas, il existe des *chemins infaisables*, c'est-à-dire des chemins du graphe qui ne correspondent pas à une exécution du programme. Ainsi, dans notre exemple, tout chemin passant par $I7$ *et* par $I9$ est infaisable. En toute généralité, déterminer qu'un chemin est infaisable est un problème indécidable. Dans notre cas particulier, il est décidable car le nombre de contraintes et le domaine des entrées sont finis. Cependant, le temps de décision peut s'avérer déraisonnable. Ce problème épineux est commun à toutes les méthodes de test statistique où le choix des données s'effectue via le choix d'un chemin d'exécution. En pratique, dans notre cas, la procédure de résolution de contraintes est stoppée si le temps dépasse une certaine limite. Les chemins infaisables ont un autre effet indésirable : ils faussent les probabilités de passage par les éléments. La qualité de test peut s'en trouver dégradée.

Nous avons jusqu'à présent adopté une simple méthode de rejet pour gérer les chemins infaisables et les chemins «non déterminés». La démarche a été appliquée avec succès sur des petits programmes. Elle est en passe d'être testée à plus grande échelle. Le problème qui nous préoccupe en premier lieu est celui de la complexité de la méthode de rejet. Si les chemins infaisables sont en trop grand nombre, elle sera inefficace. Nous envisageons de faire appel à des traitements complémentaires. Ainsi, certains ensembles de chemins infaisables peuvent être déterminés et spécifiés par une analyse statique du programme, ou bien lors de l'étape de résolution de contraintes. Le graphe de contrôle peut être modifié en conséquence pour éviter d'engendrer ces chemins. D'autre part, certaines conditions de faisabilité peuvent être exprimées par des contraintes algébriques (non rationnelles). Ainsi, passer d'un formalisme rationnel à un formalisme algébrique permettrait de réduire sensiblement le nombre de chemins infaisables engendrés.

L'algorithme de choix de l'ensemble S est aussi un point que nous devons améliorer. Nous ne savons pas encore si c'est effectivement un problème difficile, ou s'il peut être résolu polynomialement. Dans tous les cas, nous pensons que l'utilisation de méthodes d'optimisation combinatoire pourrait fournir de bonnes solutions. Nous avons choisi pour l'instant de tirer uniformément les éléments dans l'ensemble S. Il est possible que la qualité de test puisse être encore améliorée en les tirant selon une loi non uniforme bien choisie. Une réflexion dans le même sens peut être entamée en ce qui concerne la loi de génération des chemins du graphe.

A plus long terme, nous envisageons d'appliquer la génération de structures combinatoires à d'autres approches du test statistique.

Chapitre 2

Modèles de génomes et séquences aléatoires.

2.1 Introduction.

La seconde moitié du vingtième siècle a vu l'avènement de la biologie moléculaire, la science qui permet d'appréhender, à l'échelle des molécules, les mécanismes de l'expression et de la transmission de l'information génétique des êtres vivants. Depuis lors, et particulièrement après la naissance de l'ingénierie génétique dans les années soixante-dix, les progrès de cette discipline ont permis d'engranger un grand nombre de connaissances concernant le fonctionnement des génomes. Cependant, si les principaux mécanismes sont compris dans leur généralité, nous sommes encore très loin d'avoir une vision fine des constituants des génomes et de leurs interactions au sein de la dynamique cellulaire. La génomique est la science dont le but est de déterminer le contenu génétique de tout génome, son organisation, les relations entre structure et fonction de l'information qui s'y trouve, et les processus qui président à son expression. C'est un travail éminemment pluridisciplinaire dans lequel l'informatique fondamentale et les statistiques ont une place importante.

La première étape obligée de l'analyse d'une séquence génomique est l'inventaire aussi exhaustif que possible des informations qu'elle contient. S'appuyant sur les données disponibles, des modèles permettent de représenter les différents types de signaux que l'on peut trouver dans les séquences. Certains modèles sont utilisés dans les logiciels d'analyse pour prédire la présence d'informations particulières –par exemple des gènes– dans les séquences. Ils sont conçus à l'aide les données connues sur le type d'informations que l'on cherche. D'autres ont une vocation exploratoire : ils ont pour but d'aider à déterminer ce qui, dans séquences, différencie les signaux biologiquement pertinents de ceux qui participent au bruit de fond. Dans ce cas, la nature et les propriétés de ce que l'on recherche ne sont pas connues *a priori*. Le principe consiste à comparer les séquences biologiques à celles que l'on attendrait «par hasard» : des différences mises en évidence, on peut inférer des propriétés purement biologiques.

Les travaux que j'effectue dans le domaine de la génomique sont de deux types différents, mais étroitement liés. D'une part j'étudie la conception et l'utilisation de ces modèles exploratoires, d'un point de vue algorithmique ou analytique selon le cas. D'autre part, je participe à des projets d'exploration et d'analyse des génomes qui rassemblent des biologistes moléculaires, des bioinformaticiens, des mathématiciens et des informaticiens de plusieurs sensibilités. Le présent chapitre présente ces deux facettes et ce qui les lie.

Ce mémoire s'adresse particulièrement aux informaticiens ; c'est pourquoi il me semble nécessaire de présenter dans les deux premières sections de ce chapitres des bases de biologie moléculaire, puis l'état actuel des connaissances en analyse des génomes. Les modèles de séquences occupent les sections 2.4 et 2.5. C'est dans cette dernière qu'est présentée ma contribution à l'étude des modèles. Enfin, la section 2.6 est consacrée à deux études bioinformatiques pluridisciplinaire en cours à Orsay, dans lesquelles les modèles exploratoires ont une place, parmi d'autres concepts et techniques informatiques.

2.2 Bases de biologie moléculaire.

Il ne s'agit pas ici de faire un cours de biologie moléculaire –la place, et les compétences manquent– mais d'une part de présenter les mécanismes de base de l'expression de l'information génétique, d'autre part de montrer quelques points d'importance qui participent à la complexité, parfois négligée à tort, de ces mécanismes. Beaucoup de notions abordées se trouvent dans tout bon livre d'initiation à la biologie cellulaire et moléculaire (je recommande vivement l'excellent ouvrage de Jean-Claude Callen [44].) D'autres connaissances, moins «standard», ont été glanées au cours de conversations et de séances de travail avec des biologistes (particulièrement à l'IGM).

2.2.1 Macromolécules et mécanismes de base.

Les êtres vivants sont divisés en deux grandes familles [1] : les procaryotes, organismes unicellulaires ; les eucaryotes, organismes dont les cellules ont un noyau qui renferme l'information génétique (tous les êtres vivants pluricellulaires sont des eucaryotes). Dans tous ces organismes, les trois types de macromolécules qui portent ou permettent d'exprimer l'information génétique d'une cellule sont les deux acides nucléiques (ADN : acide désoxyribonucléique et ARN : acide ribonucléique) et les protéines. Schématiquement, l'ADN est la molécule qui porte l'information génétique ; l'ARN et les protéines sont les acteurs qui vont permettre à cette information de s'exprimer pour faire fonctionner la machinerie cellulaire. L'information génétique s'exprime au travers de deux mécanismes essentiels, la *transcription* et la *traduction*.

L'ADN.

L'ADN est une macromolécule constituée de molécules appelées *nucléotides*. Chaque nucléotide est lui-même composé d'une *base*, d'un sucre (désoxyribose) et d'un groupe phosphate. Il existe quatre bases dans l'ADN : *adénine* (`A`), *cytosine* (`C`), *guanine* (`G`) et *thymine* (`T`). La molécule d'ADN est composée de deux séquences de nucléotides (appelées *brins*) liées l'un à l'autre, grâce à deux types différents de liaisons chimiques qui peuvent exister de nucléotide à nucléotide :

- Une liaison covalente entre le sucre d'un nucléotide et le groupe phosphate d'un autre nucléotide. C'est elle qui permet la constitution des brins.
- Des liaisons hydrogène entre les bases elles-mêmes. Ainsi, les bases A et T peuvent être liées par deux liaisons hydrogène, tandis que les bases C et G sont susceptibles d'être liées par trois liaisons hydrogène. Ainsi, chaque A (resp. C) d'un brin se trouve en face d'un T (resp. G) du brin opposé. De cette façon, les deux brins sont *complémentaires*.

1. Les virus forment une classe à part.

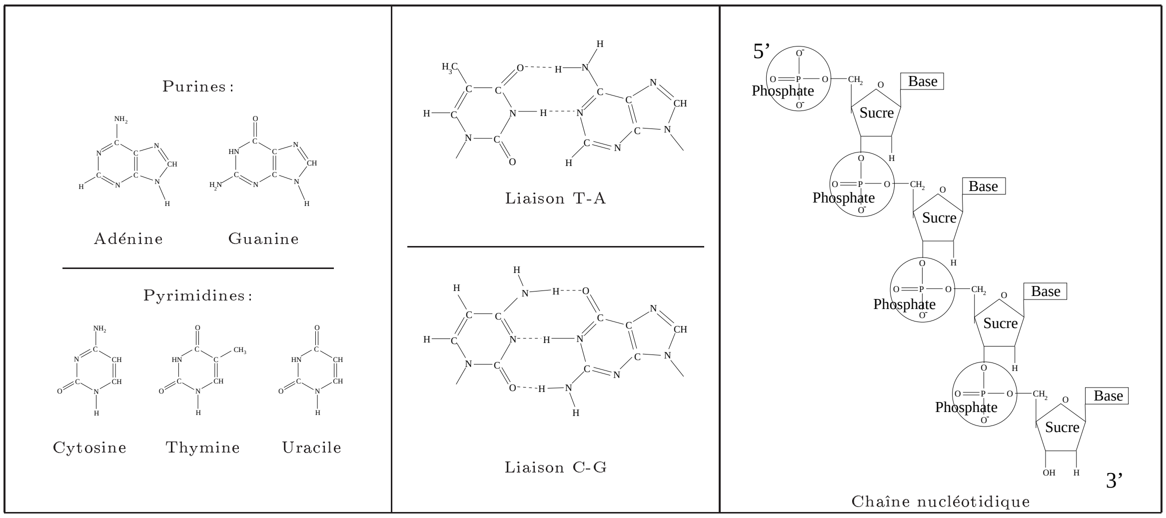


FIG. 2.1 – *Nucléotides et acides nucléiques (dessins d'après [135])*

Ces deux types de liaisons chimiques sont d'essences radicalement différentes. En particulier, les liaisons hydrogène sont des liaisons faibles, dont la rupture nécessite beaucoup moins d'énergie que celle des liaisons covalentes. La liaison covalente est asymétrique : le groupe phosphate relie le radical hydroxyle situé en position 5' d'un sucre au radical hydroxyle situé en position 3' du sucre adjacent (voir figure 2.1) [2]. Par convention, le brin est orienté dans le sens 5'-3' (cette orientation est liée au sens de la transcription de l'ADN).

La transcription et l'ARN.

La transcription est le mécanisme qui permet de copier une portion d'ADN en ARN. Une molécule d'ARN est composée, comme l'ADN, de quatre types de nucléotides différents, dont les bases sont l'adénine, la cytosine, la guanine et l'*uracile* (`U`) qui remplace la thymine. L'ARN est généralement simple brin. Seules certaines régions de l'ADN peuvent être transcrites en ARN. Ce sont ces zones qui sont communément appelées *gènes* [3]. En amont (généralement) de chaque gène se trouve une *zone promotrice*, susceptible d'accueillir une protéine particulière, l'ARN-polymérase, qui recopie, nucléotide par nucléotide, la région d'ADN concernée en un brin d'ARN.

La traduction et les protéines.

Certains ARN participent directement au fonctionnement cellulaire. D'autres –la plupart– sont traduits en protéines, acteurs majoritaires de l'activité cellulaire. Les molécules d'ARN destinées à la traduction en protéines subissent tout d'abord un processus de *maturation des transcrits*. Chez les eucaryotes, et particulièrement les eucaryotes supérieurs, une étape importante de ce processus est l'*épissage* : une grande majorité des gènes protéiques des eucaryotes supérieurs ont la particularité d'être séparés en plusieurs parties sur le brin d'ADN

2. $5'$ et $3'$ sont des numéros attribués par les chimistes aux atomes des molécules.

3. Il n'existe pas de définition précise de la notion de gène en biologie moléculaire. Cela vient probablement du fait que, pendant longtemps, le support moléculaire des gènes est resté inconnu des généticiens ; la notion de gène était alors liée aux caractères d'un organisme. La définition donnée ici semble être l'objet d'un relatif consensus chez les bioinformaticiens.

qui les porte. Ces parties s'appellent les *exons*, et elles sont séparées par des *introns*. L'ARN transcrit, qu'on appelle *ARN pré-messager* ou *transcrit primaire*, contient les exons et les introns. Au cours de la maturation, les introns vont être *excisés* et le processus d'épissage va rabouter les exons pour former le brin d'*ARN messager* (*mRNA*), qui contient la séquence ininterrompue qui sera traduite en protéine.

Les protéines sont aussi des molécules séquentielles (ou des assemblages de telles molécules). Leur composition chimique diffère de celle des ADN et ARN : une séquence protéique (ou peptide) est une succession de molécules appelées *acides aminés*. Il existe vingt acides aminés différents. La correspondance entre ARN et peptide est définie par le *code génétique* : chaque acide aminé est codé par une suite de trois nucléotides de l'ARN, appelée *codon*. Le code génétique est dit dégénéré, car plusieurs codons peuvent correspondre à un même acide aminé. La traduction d'un ARN messager est effectuée par le *ribosome*, un assemblage de macromolécules de protéines et d'ARN. Le ribosome parcourt séquentiellement le brin d'ARN dans le sens 5'-3'. La partie codante de l'ARN commence à un codon particulier, appelé codon START (généralement `AUG`). Les nucléotides sont parcourus codon par codon (sans chevauchement) ; pour chaque codon, un acide aminé est ajouté à la protéine en cours de construction. Les acteurs essentiels de ce processus sont des ARN particuliers appelés *ARN de transfert* (*tRNA*). Ils constituent l'interface entre le monde des acides nucléiques et celui des protéine. Un ARN de transfert a la particularité de pouvoir être lié chimiquement à un acide aminé précis parmi les vingt possibles. En outre, sa séquence contient un motif de trois nucléotides appelé *anticodon*, qui est complémentaire à un codon correspondant à ce même acide aminé. C'est lorsque l'ARN de transfert, au sein du ribosome, se lie au codon correspondant à son anticodon que l'acide aminé est ajouté au peptide. La traduction s'interrompt lorsque le ribosome rencontre l'un des trois codons STOP, qui ne codent pour aucun acide aminé : `UAA`, `UAG` ou `UGA`.

Régulation de l'expression des gènes.

La transcription est *régulée* par l'action de protéines qui, en interagissant avec des sites de la *région régulatrice* de la zone à transcrire, activent l'action de l'ARN-polymérase ou l'inhibent, par exemple en empêchant sa fixation. La régulation transcriptionnelle est un phénomène très complexe qui peut mettre en jeu un grand nombre de protéines qui se fixent sur des zones parfois éloignées du gène régulé. D'autres mécanismes de régulation interviennent à d'autres niveaux, par exemple pendant la traduction.

2.2.2 Structures tridimensionnelles.

Structures d'ARN et de protéines.

Les fonctions des protéines sont multiples : catalyse (enzymes), structure (membranes, fibres musculaires...), communication (hormones), protection (anticorps) etc. Ceci est vrai aussi pour certains ARN. La ou les fonctions d'une protéine ou d'un ARN donné est liée à sa structure tridimensionnelle. Celle-ci, dans beaucoup de cas, ne dépend que de la *structure primaire*, c'est-à-dire de la séquence des acides aminés ou des nucléotides qui composent la molécule.

La structure des ARN dépend principalement des liaisons hydrogène `A-U`, `G-C` et `G-U` (liaison nouvelle par rapport à l'ADN). Ces liaisons conduisent le brin à se replier dans l'espace pour privilégier les appariements, de façon à se trouver dans une configuration de moindre énergie. Cette structure spatiale est la *structure tertiaire* de la séquence d'ARN considérée. On appelle *structure secondaire* d'une séquence d'ARN une représentation planaire des appariements de sa structure tertiaire. Cette représentation est nécessairement partielle, car la totalité des appariements ne peut être représentée dans un dessin planaire.

La structure tertiaire d'une protéine est déterminée aussi en grande partie par sa séquence. Le caractère hydrophile ou hydrophobe de chacun des acides aminés semble déterminant pour le repliement. Beaucoup de protéines sont en fait constituées de plusieurs peptides liés par des liaisons non nécessairement covalentes. Il existe aussi des complexes macromoléculaires comportant aussi bien des ARN que des protéines. C'est le cas des ribosomes.

Structures d'ADN.

Il est bien connu que la molécule d'ADN, vue en trois dimensions, adopte une forme en double hélice caractéristique. Cette image, si elle est vraie à petite échelle pour de l'ADN «nu», doit être complexifiée lorsqu'on s'intéresse à la conformation de l'ADN dans la cellule et à ses interactions avec les autres macromolécules. Ainsi, la double hélice, si elle n'est soumise à aucune contrainte extérieure, n'est pas rectiligne mais adopte une conformation tridimensionnelle qui dépend des contraintes physico-chimiques des nucléotides qui la composent. D'autre part, *in vivo*, l'ADN se trouve en contact permanent avec d'autres molécules, principalement des protéines. Par exemple, chez les eucaryotes (organismes dont les cellules possèdent un *noyau*), la majeure partie de l'ADN est «enroulée» autour de protéines appelées *histones*, pour former la *chromatine*. Ce processus est dynamique et réversible : des zones de la chromatine peuvent se trouver temporairement déroulées, en fonction de certains stimuli. Il est important de retenir cette vision structurale de l'ADN, car les molécules qui déclenchent l'expression de l'information génétique ne peuvent se fixer que sur des sites accessibles et «compatibles» avec leurs structures tertiaires.

2.2.3 Variabilité et complexité de l'information génétique.

Éléments de variabilité des génomes.

La *réplication* de l'ADN est un processus essentiel dans la reproduction des êtres vivants. Il consiste à recopier à l'identique le patrimoine génétique d'un organisme pour le transmettre à sa descendance. Tous les processus moléculaires sont sujets à des erreurs qui peuvent induire des modifications dans l'ADN d'un organisme. Certaines d'entre elles sont corrigées par des mécanismes d'auto-réparation. Dans le cas contraire, on aboutit à une mutation. Une mutation peut être très ponctuelle (par exemple modification, insertion ou suppression d'une unique paire de bases), ou bien affecter des régions entières d'un chromosome. Certaines mutations sont «silencieuses», d'autres modifient les caractères de l'organisme. Cela peut être le cas lorsqu'elles touchent une région de l'ADN codant pour une protéine. Par exemple, le code génétique est ainsi fait que l'insertion ou la suppression d'un seul nucléotide dans un exon suffit à modifier totalement la séquence d'acides aminés en aval, et donc la structure de la protéine. Lorsque la mutation n'est pas létale, elle est transmise, par le processus de réplication de l'ADN, à la descendance de l'organisme concerné. L'effet d'une mutation est «jugé» opérationnellement : les êtres sujets à la mutation seront «sélectionnés» si le caractère qu'elle leur confère est neutre, ou peut leur donner un quelconque atout (vital ou reproductif) dans les conditions du milieu, ou dans des conditions ultérieures. Selon la théorie de l'évolution, ce processus d'évolution par mutation/sélection, qui a commencé il y a environ 4 milliards d'années à partir d'une ou plusieurs cellules ancestrales, a donné naissance à l'immense variété des organismes vivant ou ayant vécu sur la terre.

Nature, variété et superposition de l'information.

Chez les procaryotes, les gènes constituent la presque totalité du contenu du génome. La situation est très différente en ce qui concerne les eucaryotes, surtout chez les eucaryotes supérieurs. Par exemple, chez les mammifères, les gènes occupent seulement 5% de la longueur du génome. Les zones non géniques sont encore mal connues, mais de plus en plus on considère qu'elles ont une importance dans le fonctionnement du génome. Elles contiennent, nous l'avons vu, des zones de régulation essentielles pour l'expression des gènes. On y trouve des transposons (éléments d'ADN mobiles) et des régions très répétées. Ces deux types d'éléments jouent un rôle important dans l'évolution des génomes.

Une particularité des séquences génomiques est la superposition des informations qu'elles contiennent [175]. Par exemple, deux gènes différents peuvent se recouvrir partiellement sur la séquence d'ADN, soit sur le même brin, soit d'un brin à l'autre. D'autres mécanismes, comme l'épissage alternatif ou le recodage (section 2.6.2) par exemple, permettent de synthétiser plusieurs protéines différentes à partir d'un même transcrit. La séquence d'un ARN messager contient le code de la protéine correspondante, mais aussi, superposés à ce code, des éléments qui vont influer sur la vitesse de la traduction, et par là même sur sa régulation. L'ADN contient des informations liés à sa structuration dans la chromatine [12, 17, 177]. La stabilité de l'appariement des deux brins est capitale dans certaines régions ; elle est obtenue par une forte proportion de nucléotides `G` et `C` dans ces zones. Ainsi, chaque nucléotide peut participer à plusieurs «messages» de natures différentes.

Éléments quantitatifs.

Les tailles des génomes des organismes vivants s'échelonnent entre $6 \cdot 10^5$ paires de bases, pour certains parasites procaryotes, et $1{,}3 \cdot 10^{11}$ paires de base pour certains plantes et certains poissons. Le génome humain comporte environ $3{,}3 \cdot 10^9$ paires de bases (en fait deux fois plus, car chaque chromosome est présent en deux exemplaires). Le nombre de gènes des procaryotes les plus simples est d'environ 750. On estime que 50000 gènes pour les eucaryotes les plus complexes est une valeur limite (le génome humain en contiendrait de 30000 à 40000). Le nombre de protéines différentes produites peut être beaucoup plus élevé car un même gène peut donner plusieurs protéines, par exemple par épissage alternatif (mécanisme par lequel des exons peuvent être excisés) ou par un mécanisme de recodage (section 2.6.2). On estime à plus de 100000 le nombre de protéines différentes chez l'être humain.

Le nombre d'acteurs présents simultanément dans une cellule est extrêmement élevé. Typiquement, dans une bactérie, on observe simultanément environ $3{,}6 \cdot 10^6$ protéines et $4{,}6 \cdot 10^5$ molécules d'ARN. Une cellule d'eucaryote en contient environ 1000 fois plus.

2.2.4 Bilan.

Le but ultime des biologistes moléculaires et bioinformaticiens des génomes est d'aboutir à une compréhension fine des mécanismes qui régissent le patrimoine génétique. Bien qu'un certain nombre de processus soient bien compris aujourd'hui, nous sommes très loin du but. Ainsi, nous ne savons même pas reconnaître à coup sûr, autrement que par expérimentation biologique, la présence d'un gène dans une séquence d'ADN. La difficulté du problème est due –entre autres– à l'extrême variabilité des êtres vivants, à l'imbrication des différents messages, à l'importance des propriétés structurelles et dynamiques des séquences, au nombre d'acteurs en cause. En particulier, il est évident que la vision «classique» de l'ADN en tant que long mot sur un alphabet à quatre lettres est considérablement réductrice. Il est vrai que ce modèle –puisque c'en est un– a son utilité : la plupart des approches d'analyse à grande échelle des génomes, largement utilisées par la communauté des biologistes, ne se basent que sur cette séquence primaire ; et des avancées pourront certainement encore être faites dans le domaine, même en ne gardant que cette vision textuelle. Cependant, pour rendre compte de la complexité de l'information des séquences, il faut des modèles évolués qui prennent en considération d'autres aspects des génomes, en particulier leurs propriétés structurelles et dynamiques.

2.3 Analyse à grande échelle des génomes.

L'analyse *in silico* des génomes nouvellement séquencés consiste principalement à faire un inventaire des gènes et de les catégoriser selon leurs fonctionnalités, dans la mesure du possible. Les gènes constituent la partie la plus facile à identifier de l'information génomique. D'autre part, l'étude des fonctions des gènes protéiques débouche sur des applications à court terme, notamment en médecine et dans l'industrie.

2.3.1 Prédiction des gènes.

C'est pour les eucaryotes supérieurs que le problème est le plus difficile, du fait de la faible proportion d'ADN génique et du caractère discontinu des gènes (présence d'introns) dans les chromosomes. Une analyse correcte doit permettre de déterminer tous les exons de chaque gène. Les grandes lignes des approches d'analyse *in silico* des grands génomes récemment séquencés [9, 52, 109] sont identiques. On peut en trouver une synthèse récente dans [169].

En ce qui concerne la recherche des gènes protéiques, deux approches totalement différentes sont utilisées conjointement. La première fait appel à des algorithmes d'*alignement*, et se décline elle-même de deux façons. D'une part, on cherche à aligner sur le génome des séquences d'ARN messagers (EST : Expressed Sequence Tags ou cDNA : ADN complémentaire) extraits de cellules de l'organisme considéré. Comme ces ARN ne contiennent plus d'introns, ils doivent s'aligner de façon discontinue sur le génome, indiquant par là les positions des exons. D'autre part, le travail d'alignement est effectué avec les gènes connus d'autres organismes. Un grand nombre de gènes sont effet conservés d'un organisme à l'autre. Ce sont des gènes qui ont évolué à partir d'un même gène ancestral, et que l'on qualifie d'*homologues*. Leurs séquences sont souvent similaires car les protéines correspondantes doivent conserver des structures proches pour garder des fonctionnalités identiques.

La seconde approche, parfois qualifiée d'*ab initio*, est principalement basée sur des différences statistiques observées entre séquences codantes et séquences non codantes. On a observé depuis longtemps que les fréquences d'apparition d'oligonucléotides (c'est-à-dire de courts mots sur l'alphabet des nucléotides) varient selon que l'on se trouve dans une zone codante ou non. En vertu de ce fait, des algorithmes, souvent à base de *modèles de Markov cachés* (*HMM*), segmentent les séquences selon les zones statistiquement différentes. (On y reviendra plus longuement en section 2.4.) D'autres paramètres d'ordre statistique sont pris en compte (longueurs moyennes des introns et des exons connus, biais de codon etc.). La recherche de sites particuliers tels que les sites d'épissage, les promoteurs, les sites de polyadénylation, etc., donne des indices complémentaires.

Parmi les quelques familles d'ARN qui ne sont pas traduits en protéines, les ARN de transfert sont particuliers en ce sens que leur structure secondaire en forme de trèfle est bien connue et très conservée dans tous les organismes. Des méthodes *ad hoc* tirant parti de cette structure sont utilisées pour leur recherche. Les autres gènes à ARN sont détectés, en général, par comparaison avec des gènes connus.

Ces différentes méthodes sont couramment employées aussi pour les organismes procaryotes et eucaryotes inférieurs. Chez les procaryotes, les gènes occupent la quasi totalité du génome. Chez les eucaryotes inférieurs, les zones intergéniques occupent une proportion non négligeable mais les introns sont rares. La tâche est donc plus facile, au point que des critères très simples suffisent parfois à prédire des gènes. Ainsi, lors de l'analyse du chromosome III de la levure *S. cerevisiae* par Oliver et al. [134], était décrétée gène protéique toute séquence commençant par un codon START et comportant au moins trois cent nucléotides avant un codon STOP en phase. Les résultats ne se sont pas révélés si mauvais ! Cela dit, les modèles à base de HMM sont bien plus fiables actuellement.

2.3.2 Évaluation des prédictions.

La recherche de gènes par alignement souffre d'un défaut évident : elle ne permet aucunement de détecter des gènes nouveaux. Il n'est pas possible, en alignant des séquences d'un organisme donné avec des gènes d'autres organismes, de détecter des gènes spécifiques à l'espèce. Ainsi, chez l'homme, moins de la moitié des gènes prédits l'ont été par ce moyen [52, 109]. La comparaison avec les ARN messagers pose un problème similaire : ne peuvent être pris en compte que ceux qui sont présents dans l'organisme lors de l'extraction. Ceux-ci ne constituent qu'une fraction des ARN messagers possibles. On sait par exemple que, dans des conditions expérimentales habituelles, c'est-à-dire en phase de croissance exponentielle, seul un quart des gènes de la bactérie *Escherichia coli* sont transcrits [14]. Parmi ceux-ci, certains sont trop peu exprimés pour être détectés. D'autre part, des erreurs d'analyse peuvent survenir pour des raisons liées à la technique d'extraction des ARN messagers (contamination par des ARN pré-messagers par exemple), ou à l'imprécision des méthodes d'alignement (certains gènes évoluent rapidement, de sorte que les homologies ne sont pas décelables par les algorithmes d'alignement actuels).

Dans ces conditions, seule l'approche *ab initio* semble valide pour découvrir des gènes nouveaux. Cependant, plusieurs études récentes révèlent ses insuffisances [14, 136, 144]. Parmi les critères d'évaluation de la qualité des résultats des logiciels d'analyse, la *sensitivité* et la *spécificité* sont des plus pertinents [43, 102]. Considérons qu'un logiciel d'analyse a pour but de classer des éléments selon qu'une certaine propriété P soit vraie ou non. Les éléments positifs sont ceux qui obéissent à P, les négatifs ceux qui n'y obéissent pas. La sensitivité d'une analyse est égale à la proportion d'éléments positifs effectivement prédits positifs. La spécificité est la proportion d'éléments prédits positifs qui sont effectivement positifs. Bien entendu, ces valeurs sont calculées sur un ensemble déterminé d'éléments dont on connaît la qualité positive ou négative. Dans l'analyse des génomes, on peut prendre chaque nucléotide de la séquence à analyser comme élément, et calculer la sensitivité et la spécificité sur la propriété de chacun d'eux d'être codant (c'est-à-dire d'appartenir à un exon) ou non. Ces deux quantités s'appliquent de la même façon à la prédiction des exons ou des gènes, en considérant qu'un exon (ou un gène) prédit est correct s'il est absolument identique à l'exon (ou au gène) réel. Cette identité exacte est essentielle, car une simple erreur d'un nucléotide peut occasionner un changement de phase qui modifie du tout au tout la séquence de la protéine produite. Plusieurs programmes d'analyse *ab initio* ont récemment été évalués sur des séquences d'*Arabidopsis thaliana*. D'après [136], le meilleur programme pour la prédiction des exons est GeneMark.hmm [124], qui présente une sensitivité de 0,82 et une spécificité de 0,77. Toutefois, l'assemblage des exons est encore très problématique : GeneMark.hmm n'est capable de reconstituer que 67 gènes sur les 168 présents dans le jeu de séquences ; et le meilleur programme testé dans [144] (Genie [143]) présente une sensitivité de 0,40 et une spécificité de 0,29 en ce qui concerne la prédiction des gènes entiers. Cela signifie que moins d'un gène sur deux est correctement prédit et que les deux tiers des gènes prédits sont incorrects.

2.4 Modèles prédictifs.

En première approximation, nous dirons qu'un modèle est une représentation simplifiée d'une réalité. En ce sens, la représentation d'un chromosome par la suite des lettres de l'un de ses brins est un modèle, comme nous l'avons vu en section 2.2.4. Convenons de l'appeler *modèle alphabétique*. Un chromosome peut aussi être modélisé de façon différente, par exemple par des *cartes de courbure*, comme nous le verrons en section 2.6.1. De tels modèles sont destinés à donner de la séquence une représentation assez simple pour qu'elle puisse être étudiée et manipulée en tant qu'objet formel. Pour cette raison, nous les appellerons *modèles descriptifs*. Toute méthode d'étude *in silico* doit s'appuyer sur un ou plusieurs modèles descriptifs de la séquence à analyser. Pour la quasi totalité des méthodes d'analyse à grande échelle des génomes, il s'agit du modèle alphabétique uniquement.

Analyser une séquence *in silico* consiste à la segmenter en zones et à classer chaque zone dans une catégorie donnée. Par exemple, un outil d'analyse des génomes eucaryotes peut comporter cinq classes : intergénique, intron, premier exon d'un gène, exon interne, dernier exon. La segmentation s'effectue par rapport à un *modèle prédictif*, qui contient les données et les traitements utilisés pour la reconnaissance de chaque zone. Un bon modèle prédictif doit contenir toutes les propriétés connues du type d'information que l'on cherche, dans une représentation des données adaptée à un traitement rapide de la séquence. Cette section passe en revue les principaux types de modèles actuellement utilisés.

2.4.1 Modèles syntaxiques.

Il existe une grande variété de modèles pour représenter des *motifs*, c'est-à-dire des ensembles de mots, de façon déterministe ou probabiliste [42]. Ces modèles sont employés pour repérer des signaux conservés d'un site à l'autre (comme des éléments de promoteur, par exemple), mais qui peuvent être sujets à des variations. Le modèle le plus simple est le *consensus*, qui représente un ensemble de mots de même longueur (figure 2.2 (a)). Des généralisations de ce modèle permet de prendre en compte des séquences de taille variable. C'est le cas du format PROSITE, utilisé essentiellement pour les protéines : la base PROSITE [108] contient des centaines de descriptions de motifs permettant de classer des protéines par familles fonctionnelles. Un motif PROSITE ne permet pas de représenter les langages infinis (ce n'est pas gênant dans le cadre de son utilisation). A cette (importante) différence près, un tel motif a la même expressivité qu'une expression rationnelle. En revanche, dans le cadre de son utili-

```
CTAAAAATAA
TTAAAAATAA
TTTAAAATAA
CTATAAATAA
TTATAAATAA
CTTAAAATAG
TTTAAAATAG

YTWWAAATAR
```

(a)

	1	2	3	4	5	6	7	8	9	10
A	–	–	1.39	1.61	1.95	1.95	1.95	–	1.95	1.61
C	1.10	–	–	–	–	–	–	–	–	–
G	–	–	–	–	–	–	–	–	–	0.69
T	1.39	1.95	1.10	0.69	–	–	–	1.95	–	–

(b)

FIG. 2.2 – *(a) Un ensemble de séquences et le consensus qui le représente (séquences MEF2 [193].) Le symbole* W *(pour « Weak») signifie «*A *ou* T*» (bases à deux liaisons hydrogène seulement) ; le symbole* R *représente les purines,* A *ou* G *(voir figure 2.1). (b) Une matrice de positions pondérée pour le même ensemble. Le score d'un nucléotide est le logarithme de son nombre d'occurrences à la position considérée. Les tirets indiquent des scores non définis (donc des nucléotides interdits).*

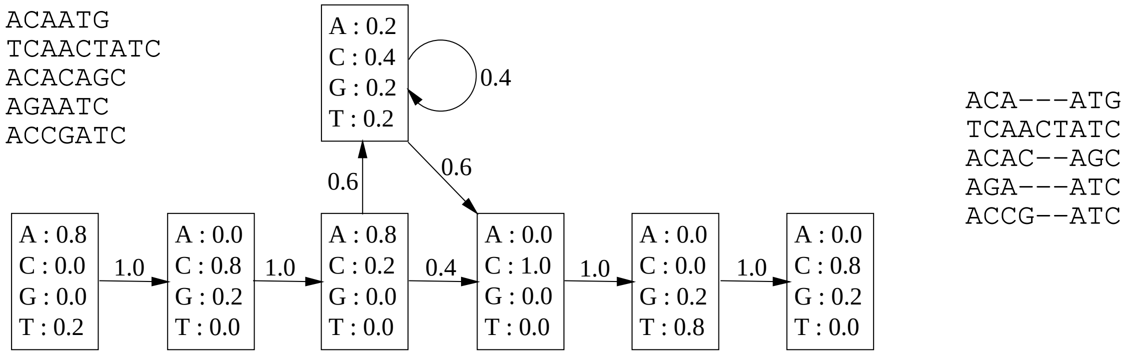
```
    C-x(4,6)-[FYH]-x(5,10) -C-x(0,2)-C-x(2,3)-C-x(7,11) -C-x(4,6)-[DNEQSP]-x(2)-C
(a) |           |             |        |        |           |           |           |
    C--RKNQ----Y-RHYWSENLFQ-C---FN---C---SL---C-LNGTVHLS-C--QEKQ----N-------TV--C


    C-x(4,6)-[FYH]-x(5,10) -C-x(0,2)-C-x(2,3)-C-x(7,11) -C-x(4,6)-[DNEQSP]-x(2)-C
(b) |            |            |        |        |           |           |           |
    C--RKNQYR---H--YWSENLFQ-C---FN---C---SL---C-LNGTVHLS-C--QEKQ----N-------TV--C
```

FIG. 2.3 – *(d'après [42]). Deux alignements différents (a et b) entre un motif PROSITE et une séquence d'acides aminés. La notation* x(*i*,*j*) *signifie que toute séquence de longueur comprise entre i et j est autorisée à cet endroit. Un ensemble de lettres entre crochets signifie qu'à la position considérée, seules les lettres de l'ensemble sont autorisées.*

sation, un motif PROSITE s'écrit en général plus simplement qu'une expression rationnelle équivalente. Tout comme une expression rationnelle, un motif PROSITE peut être ambigu : un même mot peut être aligné de plusieurs façons différentes sur le même motif. Chaque alignement peut donner lieu à une hypothèse biologique différente. La figure 2.3 en présente un exemple.

Les modèles précédents sont déterministes, en ce sens que tout mot donné obéit au modèle, ou bien n'y obéit pas. Une façon de les enrichir est d'y ajouter une notion quantitative de «proximité» : à tout mot peut être associé un score obtenu par comparaison avec le modèle. Plus le score est élevé, plus la probabilité que le mot obéisse au modèle est grande. L'équivalent probabiliste du consensus est la *matrice de positions pondérée*, définie dans [166] (figure 2.2) : à chaque position dans le motif est associée une table donnant un score pour chaque lettre de l'alphabet ; le score d'un mot est la somme des scores de chacune de ses lettres sur sa position. Comme dans le cas déterministe, des généralisations permettent de gérer les mots de longueur variable. Parmi celles-ci, les *profils* [98, 99], les *profils HMM* [76] (figure 2.4)



(a) (b)

FIG. 2.4 – *(d'après [117]). (a) Un ensemble de séquences et un profil HMM qui le caractérise. Exemple : la séquence* `AGCTAGC` *a pour probabilité* $0.8 \times 1.0 \times 0.2 \times 1.0 \times 0.2 \times 0.6 \times 0.2 \times 0.6 \times 1.0 \times 1.0 \times 0.2 \times 1.0 \times 0.8 = 0.00036864$*. (b) L'alignement à partir duquel le profil a été constitué.*

et les *profils généralisés* [41, 42] sont largement utilisés (la base PROSITE contient aussi des profils généralisés de familles de protéines). De la même façon que les motifs PROSITE et les expressions rationnelles, ces modèles peuvent être ambigus ; cependant, le calcul des scores permet de déterminer le ou les alignements les plus probables selon le modèle, donc –en théorie– les plus pertinents biologiquement. Ces modèles sont tous des cas particuliers de *modèles de Markov cachés* (HMM). On peut les considérer comme des automates finis non déterministes dans lesquels une probabilité est associée à chaque transition.

D'autres modèles tiennent compte de propriétés syntaxiques plus fortes en considérant des descriptions de séquences par des langages algébriques. La classe des langages algébriques contient celle des rationnels. On sait depuis longtemps que certaines organisations syntaxiques des séquences, comme les structures secondaires d'ARN (sans pseudo-nœud), ne peuvent pas être décrites par des langages rationnels mais peuvent l'être par des langages algébriques [161, 163, 164, 182]. En effet, les langages algébriques, contrairement aux rationnels, peuvent formaliser certaines *interactions à longue distance* (par exemple, dans le cas de l'ARN, des liaisons hydrogène entre nucléotides éloignés). Dans une *grammaire algébrique stochastique* [76], chaque règle de dérivation est pondérée par une probabilité. Dans le contexte de l'analyse des génomes, cette notion généralise celle de HMM. Les grammaires stochastiques sont utilisées dans le cadre de la recherche des ARN de transfert [78, 123]. Les méthodes de prédiction de structures secondaires ou d'alignement d'ARN peuvent aussi faire appel à ce même formalisme [78, 155] ou une de ses variantes : grammaires multibandes attribuées [118, 119] ou grammaires incluant les pseudo-nœuds [150]. Dans [162], Searls définit et étudie une classe de langages plus générale que celle des algébriques pour la modélisation des séquences d'ADN : les «*string variable grammars*» permettent de représenter des particularités des séquences (pseudo-nœuds ou répétitions en tandem par exemple) qui sont hors de portée des grammaires algébriques. Un outil général de modélisation et d'analyse des séquences a été développé selon ce formalisme [72, 163].

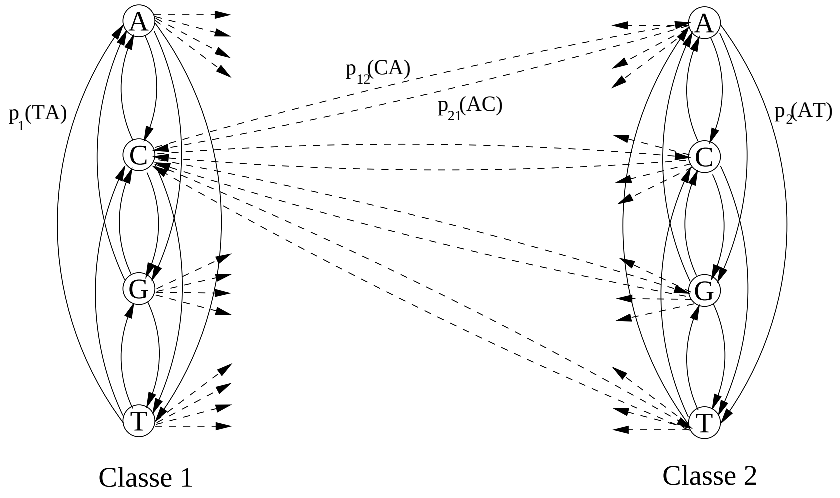


FIG. 2.5 – *Un modèle de Markov caché d'ordre 1. Pour des raisons de lisibilité, toutes les transitions et toutes les probabilités ne sont pas représentées. Exemple : au mot* `TACAT` *correspondent 32 chemins différents dans le graphe. Parmi ceux-ci, le chemin* $\mathtt{T}_1\mathtt{A}_1\mathtt{C}_1\mathtt{A}_2\mathtt{T}_2$ *détermine une segmentation dans laquelle* `TAC` *est dans la classe 1 et* `AT` *dans la classe 2. Sa probabilité est égale à* $p_1(\mathtt{TA}) \cdot p_1(\mathtt{AC}) \cdot p_{12}(\mathtt{CA}) \cdot p_2(\mathtt{AT})$.

2.4.2 Modèles basés sur la composition nucléotidique.

On sait depuis longtemps que la composition nucléotidique d'une séquence génomique n'est pas homogène : les fréquences moyennes d'apparition des quatre nucléotides varient tout au long de la séquence. Déjà en 1974 [79], Elton proposait un modèle probabiliste rendant compte de cette observation. Puis est apparue la notion de *linguistique markovienne* d'une séquence [25, 36, 45], et le fait que des linguistiques différentes d'une zone à l'autre peuvent refléter des fonctionnalités différentes (en particulier la qualité codante ou non codante d'une zone). Dans la linguistique markovienne, on s'intéresse au dénombrement des mots (chevauchants) de longueur fixée k dans les séquences. Chaque classe (par exemple codant ou non codant) est définie par sa linguistique, c'est-à-dire par les fréquences moyennes d'apparition de ces mots. Les modèles prédictifs fondés sur ces propriétés statistiques sont le plus souvent des modèles de Markov cachés. La figure 2.5 en donne un exemple avec deux classes. Les transitions sont pondérées par des probabilités qui dépendent de la linguistique de chacune des classes et des conditions de passage d'une classe à l'autre. A toute séquence de longueur n donnée en entrée correspondent un certain nombre de chemins dans le graphe[4]. La segmentation retenue est celle dont la probabilité (produit des probabilités des transitions utilisées) est maximale. Ce modèle de Markov est dit d'ordre 1, car à tout moment dans le traitement de la séquence, la probabilité d'une transition ne dépend que de la dernière lettre lue (l'ordre est toujours égal au paramètre k de la linguistique moins un). Les modèles de Markov cachés réellement utilisés sont d'ordre supérieur et plus complexes car ils tiennent compte d'autres paramètres, comme la composition codonique des zones codantes. On trouvera dans [130] une présentation des différents types de modèles, incluant des variantes des modèles de Markov. Les modèles de Markov cachés peuvent intégrer à la fois des paramètres statistiques

4. Le qualificatif «caché» provient justement du fait que, si l'on observe une séquence censée obéir au modèle, on ne peut pas *a priori* connaître la suite de transitions correspondante.

et de paramètres syntaxiques (rationnels) vus dans le paragraphe précédent (voir [117] pour une présentation générale). Il existe aussi des méthodes de segmentation selon la linguistique fondées sur d'autres modèles (voir [35] pour un état de l'art en 1998 ; [16, 100, 126] pour des travaux plus récents).

2.4.3 Autres modèles.

Dans [191, 192], une approche alternative est proposée pour la prédiction des gènes. Elle est fondée sur des propriétés thermodynamiques de la double hélice. Dans certains génomes, les régions codantes semblent être d'une grande stabilité de ce point de vue. L'originalité de cette approche est que des interactions à longue distance sont prises en compte pour évaluer la stabilité thermodynamique d'une région d'ADN.

D'autres travaux indiquent que des propriétés structurales (courbure, flexibilité, stabilité thermique) des promoteurs peuvent être conservées même lorsque la séquence primaire ne l'est pas [128, 137]. Des travaux de prédiction de zones régulatrices basés sur des modèles structuraux ont vu le jour [27, 114]. Toutefois, les avancées dans ce domaine sont encore très timides [40, 81].

2.4.4 Point de vue algorithmique.

Il est évident que la complexité des algorithmes de prédiction est un paramètre qui doit être pris en compte, étant donné la taille et la quantité des séquences à analyser. La recherche de motifs rationnels peut s'effectuer en temps et espaces linéaires [53, 54, 103]. Les prédictions à base de HMM s'effectuent en temps $O(nm^2)$ et en espace $O(nm)$, où n est la longueur de la séquence à analyser et m le nombre d'états de la chaîne de Markov. Les modèles fondés sur des grammaires algébriques stochastiques ont une complexité plus élevée, $O(n^3m^3)$ en temps et $O(n^2m)$ en espace [76, p258]. Il est bien connu que, plus on s'élève dans la hiérarchie de Chomsky, plus la complexité intrinsèque des algorithmes d'analyse syntaxique est élevée. La faculté d'expression des *string variable grammars* les situe légèrement au-dessus des grammaires algébriques. La complexité des algorithmes associés est donc probablement plus élevée, sans pour autant être obligatoirement au niveau de celle des grammaires contextuelles générales. (Il semble qu'aucune analyse de complexité relative aux *string variable grammars* n'ait été faite.) L'approche de [191, 192], fondée sur la stabilité thermodynamique et tenant compte d'interactions à grande distance, a une complexité en $O(n^2)$.

La complexité des algorithmes est très certainement un facteur limitant dans l'analyse des génomes. C'est aussi parce que les interactions à longue distance sont mal connues que peu d'outils d'analyse font appel à des langages plus expressifs que les rationnels.

2.5 Modèles exploratoires.

La conception d'un modèle prédictif doit bien évidemment s'appuyer sur des données. Les résultats expérimentaux des biologistes moléculaires fournissent des données précises sur des phénomènes particuliers. Ainsi par exemple, les motifs et profils généralisés de PROSITE sont construits, à la base, à partir d'ensembles de protéines dont on a déterminé expérimentalement qu'elles appartiennent à une même famille [42, 107]. Cependant, sauf cas particuliers, ces données expérimentales sont peu nombreuses en comparaison de la quantité d'informations à analyser et de la variabilité des phénomènes à étudier.

Une autre source importante d'informations s'appuie sur ce que j'appellerai des *modèles exploratoires*. Un modèle exploratoire est un ensemble, éventuellement infini, de séquences auquel est associée une loi de probabilité (définie explicitement ou non). Le modèle est conçu selon des propriétés particulières observées sur des séquences réelles. Les modèles exploratoires sont couramment utilisés pour aider à la découverte de caractéristiques nouvelles des séquences considérées. La démarche consiste à comparer les propriétés de séquences aléatoires obéissant à un modèle exploratoire donné avec des propriétés observées sur des séquences réelles. Des différences observées, on déduit des faits biologiques. En quelque sorte, un modèle exploratoire représente le «bruit de fond» à partir duquel, par contraste, on cherche à déterminer quels sont les éléments biologiquement pertinents dans les séquences réelles.

Les modèles exploratoires les plus courants sont dits homogènes car ils reposent sur l'hypothèse qu'une séquence est régie de bout en bout par une même loi statistique. Cette hypothèse est tout à fait raisonnable si l'on cherche à modéliser un ensemble de courtes séquences d'un même organisme ayant des fonctionnalités communes. Elle l'est moins si l'on s'intéresse à des longues séquences, comme des chromosomes entiers. Pourtant, ce modèle grossier est suffisant pour certaines applications, comme on le verra. Cependant, pour mettre en évidence des propriétés fines des séquences, il est nécessaire de disposer de modèles qui tiennent compte de propriétés locales. C'est une des raisons d'être des modèles hétérogènes.

2.5.1 De l'utilisation des modèles exploratoires.

Modéliser pour rechercher les exceptions.

Un exemple emblématique d'utilisation d'un modèle exploratoire est la recherche de motifs exceptionnels pour identifier des sites d'intérêt biologique. L'hypothèse de départ –souvent avérée– est qu'un motif exceptionnellement sur-représenté dans une séquence biologique est le signe d'une fonctionnalité biologique. Inversement, une sous-représentation peut indiquer un motif évité car néfaste pour l'organisme. La sur- ou sous-représentation est mesurée en comparant le nombre d'occurrences observées dans les séquences naturelles et celui attendu dans des séquences aléatoires obéissant à un modèle exploratoire donné. Le motif «Chi» est un exemple de motif sur-représenté dans des génomes de bactéries [130, 133]. Ces organismes possèdent un complexe protéique, RecBCD, dont le rôle est de détruire toute chaîne d'ADN étrangère. Pour que ce complexe «reconnaisse» l'ADN de l'organisme lui-même, le motif Chi, dont le rôle est d'inhiber l'action de RecBCD, est disséminé dans tout le génome. D'un autre côté, les *sites de restriction*, motifs reconnus par une *enzyme de restriction* dont la fonction est de couper la molécule d'ADN, sont des motifs qui ont tendance à être contre-sélectionnés. Dans ces deux exemples, on compte le nombre d'occurrences d'un motif dans une seule (longue) séquence. Pour d'autres applications, on considère un ensemble de séquences et on compte combien d'entre elles contiennent le motif d'intérêt. C'est le cas par exemple pour l'identification de sites de polyadénylation [24] : on détermine quels sont les hexanucléotides que l'on retrouve dans de nombreuses extrémités $3'$ d'ARN messagers. Le problème éminemment important de l'identification des zones promotrices fait appel à des méthodes similaires : on recherche des motifs souvent présents dans des zones connues pour contenir des promoteurs [153, 181].

D'autres champs de l'analyse des séquences font appel de manière similaire à des modèles exploratoires. Il en est ainsi de la recherche d'homologies par alignement : pour évaluer la pertinence biologique des scores d'alignement obtenus, on les compare à ceux obtenus sur des séquences aléatoires [8, 121, 122]). Dans [80, 170] les distributions des longueurs des gènes dans *S. cerevisiae* sont comparées à celles de séquences aléatoires. Dans [11, 13], des modèles sont conçus pour représenter et étudier des périodicités particulières observées dans les séquences. Les séquences aléatoires sont aussi employées pour tester ou étalonner des algorithmes et logiciels d'analyse et de traitement des séquences [28, 123]). Elles peuvent être partie intégrante d'algorithmes de traitement de séquences. C'est le cas dans [115], où la détection de répétitions en tandem fait appel à l'évaluation de la sur-représentation de motifs.

Modèles «réalistes» et expérimentation biologique.

Plus généralement, un modèle exploratoire est un outil pour dégager des propriétés nouvelles et formuler des hypothèses sur les séquences. La linguistique markovienne est un fruit de cette démarche : elle est née de la comparaison du nombre d'occurrences observées de certains mots dans les séquences réelles et de leur nombre attendu dans des séquences aléatoires [36]. C'est aussi en comparant les fréquences des nucléotides le long des séquences réelles et des séquences aléatoires que l'on a mis en évidence l'existence de zones de distributions nucléotidiques différentes dans les génomes [79]. De là découlent les méthodes d'analyse *ab initio* fondées sur la composition nucléotidique (section 2.4). Par ailleurs, les propriétés mises en évidence par un modèle exploratoire donné permettent de concevoir des modèles plus précis [45, 79, 82]. Ces modèles sont qualifiés d'*hétérogènes* car ils intègrent le fait que des zones différentes d'une même séquence obéissent à des lois différentes. Le but est de définir des lois ou des processus mathématiques capables de rendre compte aussi finement que possible de certaines propriétés des génomes séquencés. Jusqu'il y a peu, les modèles hétérogènes ne prenaient en compte que des contraintes d'ordre statistique sur les nombres d'occurrences d'oligonucléotides. Une partie importante de mon travail consiste à étudier la conception et l'algorithmique de nouveaux modèles dits *structurés*, car ils tiennent compte de contraintes syntaxiques en sus de contraintes statistiques classiques.

Cette modélisation de plus en plus «réaliste» est utile à plusieurs titres. La comparaison du modèle avec la réalité permet de dégager des caractéristiques plus fines des séquences, suscitant ainsi de nouvelles questions et hypothèses biologiques. Nous verrons en section 2.5.3 que la détection de motifs sur-représentés nécessite parfois la prise en compte de propriétés syntaxiques des séquences. En section 2.6.1, je présenterai un travail dans lequel un modèle exploratoire hétérogène est utilisé pour tester l'indépendance de deux types de propriétés. De plus, les modèles structurés ouvrent la voie à une manière inédite d'évaluer expérimentalement un modèle de séquences : il s'agit d'engendrer aléatoirement *in silico* des séquences obéissant au modèle[5], de les synthétiser puis de les tester *in vitro* ou *in vivo*. Cette approche est en voie d'être utilisée dans les travaux sur le décalage du cadre de lecture en traduction présentés en section 2.6.2.

Des modèles exploratoires aux modèles prédictifs.

Les modèles exploratoires ont pour vocation de donner des éléments d'information et de réflexion sur les séquences qui peuvent aboutir à la conception de modèles prédictifs. Cette

5. Il existe un procédé purement biologique de génération de séquences aléatoires. La *méthode Selex* [188] consiste à engendrer simultanément *in vitro* un très grand nombre de séquences aléatoires sans contrainte (sinon des contraintes biologiques non maîtrisées), puis de sélectionner celles qui satisfont les critères choisis (activité enzymatique particulière par exemple). On peut voir cela comme une méthode biologique de génération aléatoire de séquences d'intérêt par rejet. Cependant, on ne peut pas maîtriser les paramètres du processus de génération et la méthode n'est applicable que sur des séquences de taille limitée.

relation entre les deux types de modèles pousse à s'interroger sur ce qui les rapproche et ce qui les différencie.

Bien entendu, le côté algorithmique de la prédiction est absent d'un modèle exploratoire ; il est remplacé par des procédures de génération aléatoire ou d'étude analytique des modèles. Par ailleurs, un modèle prédictif parfait devrait contenir toutes les propriétés requises pour prédire avec une sensitivité et une spécificité de 100%. En revanche, un modèle exploratoire est, par essence, incomplet : il rend compte seulement de certaines propriétés des séquences considérées. Parfois, c'est par force : le modèle est composé de tout ce que l'on connaît ou croit connaître de propre au phénomène que l'on étudie, dans l'espoir que la démarche de comparaison ou d'expérimentation biologique amènera des éléments complémentaires. Le modèle est alors, en quelque sorte, une esquisse de modèle prédictif. Dans les autres cas, les propriétés constitutives du modèle exploratoire doivent être choisies en fonction de ce que l'on cherche à mettre en évidence. Ce choix est parfois délicat. Sa justification vient souvent *a posteriori*, selon que des propriétés nouvelles sont ou non mises en évidence par le modèle.

2.5.2 Modèles homogènes.

Modèle exact et modèle markovien.

Un modèle homogène est fondé uniquement sur des propriétés globales de la ou des séquences réelles de référence. Dans la plupart des cas, c'est la *linguistique globale* qui est prise en compte : les paramètres du modèle sont les fréquences observées des nucléotides ou oligonucléotides de longueur fixée l dans la séquence naturelle de référence. A partir de ces fréquences, deux modèles exploratoires sont possibles, dont les bases sont décrites dans [83] : le *modèle exact* et le *modèle markovien*. Les séquences obéissant au modèle markovien respectent *en moyenne* les fréquences d'oligonucléotides données. Ce modèle est défini par une chaîne de Markov d'ordre $l-1$ sur les lettres de l'alphabet N des nucléotides. Les probabilités de transition sont calculées comme suit :

$$\Pr(x|w) = \frac{\#(wx)}{\sum_{y \in N} \#(wy)}$$

où $x \in N$, $y \in N$, $w \in N^{l-1}$ et, pour tout mot $v \in N^*$, $\#v$ désigne le nombre d'occurrences de v dans la séquence de référence. Le graphe de cette chaîne de Markov est un sous-graphe du graphe de De Bruijn [152] : chaque sommet-état est représenté par un oligonucléotide de longueur $l-1$; il ne peut exister un arc d'un état u à un état v que si les $l-2$ dernières lettres de u sont identiques aux $l-2$ premières de v (figure 2.6) ; cet arc est étiqueté par la probabilité d'engendrer la dernière lettre de v sachant qu'on vient d'engendrer celles de u. Le modèle exact impose de respecter *exactement* les fréquences données. Les séquences obéissant à ce modèle doivent être de la même longueur que la séquence de référence. Un tel modèle peut être représenté par un multigraphe dont chaque sommet est étiqueté par un oligonucléotide de longueur $l-1$. De même que précédemment, le graphe sous-jacent est un sous-graphe du graphe de De Bruijn (voir figure 2.6). Le nombre d'arcs joignant deux sommets u et v est égal au nombre d'occurrences de l'oligonucléotide $u.v_{l-1}$ dans la séquence de référence (v_{l-1} désigne la dernière lettre de v). L'ensemble des séquences obéissant au modèle est en bijection avec l'ensemble des chemins eulériens du multigraphe ainsi constitué (c'est-à-dire l'ensemble des chemins passant une fois et une seule par chacun des arcs). La loi de probabilité du modèle est la loi uniforme : toutes les séquences obéissant au modèle sont équiprobables.

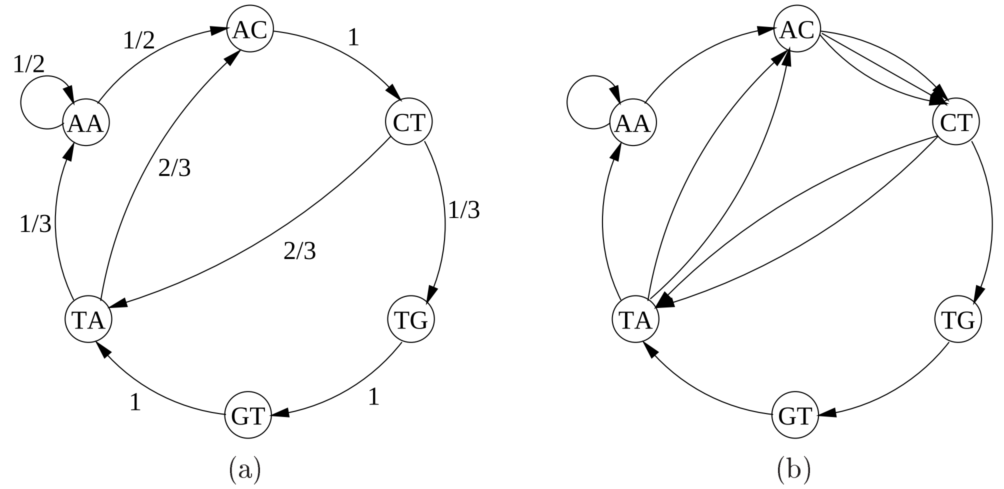


FIG. 2.6 – *Représentations graphiques des deux modèles exploratoires homogènes, pour $l = 3$ et la séquence de référence* `ACTACTGTAAACTAC`*. (a) Modèle markovien. (b) Modèle en fréquences exactes. (Le graphe est fortement connexe car la séquence commence et finit par les mêmes $l - 1$ nucléotides.)*

Remarquons que le cas $l = 1$ est très particulier. Le modèle markovien se réduit alors à une loi de Bernoulli : les lettres sont engendrées indépendamment du contexte. Le modèle exact revient à un tirage sans remise.

Il existe des raffinements et des variantes de ces modèles de base. La spécificité des séquences codantes est prise en compte dans certains modèles markoviens. Ainsi, dans [7], pour des séquences codantes, on considère un modèle qui préserve simultanément les nombres exacts d'occurrences de dinucléotides et de codons. Dans d'autres travaux, la probabilité de transition d'un état à l'autre dépend de la *phase* du nucléotide concerné[6]. Dans [11], on fait usage d'un modèle original pour rendre compte de périodicités (autres que celle associée aux codons) observées dans les gènes. Les séquences sont engendrées par un automate probabiliste[7], puis modifiées par un processus de mutations, insertions et suppressions aléatoires de nucléotides.

Algorithmes de génération de séquences aléatoires.

La génération de séquences selon le modèle markovien ne pose pas de difficulté [83] : il suffit, à chaque étape, d'engendrer une lettre x avec la probabilité donnée par la formule de la page 42, dans laquelle le mot w représente les $l - 1$ dernières lettres engendrées. Les $l - 1$ premières lettres de la séquence aléatoire peuvent être les mêmes que celles de la séquence de référence, ou bien être engendrées elles aussi en tenant compte des fréquences d'oligonucléotides. L'algorithme de génération est bien évidemment linéaire en la longueur de la séquence. Le problème est plus délicat dans le modèle exact. La bijection entre séquences et chemins eulériens du multigraphe est signalée dans [83]. Engendrer un chemin eulérien dans un graphe eulérien n'est pas difficile ; cependant la condition d'uniformité complique les

6. Dans une séquences codante, on considère trois phases de lecture, selon la position du nucléotide dans son codon ; le premier nucléotide de chaque codon est en phase 0, le second en phase 1, le dernier en phase 2.

7. qui équivaut à une chaîne de Markov ; mais celle-ci est plus complexe que dans les modèles classiques.

choses. Notons par ailleurs que, par construction, le multigraphe est chemin-eulérien, mais pas nécessairement circuit-eulérien : il existe au moins un chemin eulérien (correspondant à la séquence de référence), mais pas forcément de circuit eulérien ; et s'il n'existe pas de circuit eulérien, alors tous les chemins eulériens ont même sommet initial i et même sommet terminal t. Ceci implique que, sauf cas exceptionnel, toute séquence obéissant au modèle devra commencer et finir par les $l-1$ mêmes nucléotides que la séquence de référence. La solution au problème de génération uniforme est donnée par une relation entre les chemins eulériens et les *anti-arbres couvrants* du multigraphe. Cette relation, qui semble appartenir au folklore, est notamment décrite dans [152]. Un anti-arbre couvrant est un arbre couvrant enraciné dont tous les arcs sont orientés vers la racine. Considérons un anti-arbre couvrant du multigraphe, dont la racine est le sommet t. Construisons un chemin à partir du sommet i de la façon suivante : à chaque étape, choisir uniformément au hasard et emprunter un arc qui est adjacent au sommet courant, qui n'a pas encore été emprunté et qui n'appartient pas à l'anti-arbre. S'il n'en existe pas, emprunter l'arc de l'anti-arbre. Ce processus garantit l'obtention d'un chemin eulérien uniforme pourvu que l'anti-arbre ait aussi été engendré uniformément. Dans le premier article traitant de cette approche [7], celui-ci était engendré par une méthode de rejet qui ne permettait de résoudre raisonnablement que le cas $l = 2$. Un algorithme plus efficace a été présenté dans [113], profitant des travaux sur la génération d'arbres couvrants de [4, 39] évoqués en section 1.4. Sa complexité est en $O(q^2 n)$, où q est le nombre de sommets du graphe et n la longueur de la séquence.

2.5.3 Sur-représentation de motifs : approches analytiques.

Pour décider si un motif est exceptionnellement sur- ou sous-représenté dans une séquence, il est nécessaire de disposer d'une mesure de comparaison des observations avec une loi de probabilité théorique. Le Z-score et la p-valeur sont les mesures les plus employées. Supposons qu'un certain motif H a été observé Obs(H) fois dans une séquence naturelle de longueur n. Considérons une variable aléatoire $X_{\mathrm{H},n}$ qui décrit la loi du nombre d'occurrences de H dans une séquence aléatoire de même longueur n. Le Z-score du motif est donné par

$$Z(\mathrm{H}) = \frac{E(X_{\mathrm{H},n}) - \mathrm{Obs}(\mathrm{H})}{\sqrt{V(X_{\mathrm{H},n})}}$$

où $E(X_{\mathrm{H},n})$ et $V(X_{\mathrm{H},n})$ désignent respectivement l'espérance et la variance de $X_{\mathrm{H},n}$. La p-valeur de H est la probabilité qu'il apparaisse au moins autant de fois dans une séquence aléatoire qu'il apparaît dans la séquence naturelle :

$$\mathrm{pval}(\mathrm{H}) = \Pr(X_{\mathrm{H},n} \geq r) \qquad \text{ou bien} \qquad \mathrm{pval}(\mathrm{H}) = \frac{\Pr(X_{\mathrm{H},n} \geq r)}{\Pr(X_{\mathrm{H},n} \geq 1)} \text{ si } E(X_{\mathrm{H},n}) \ll 1.$$

La deuxième formule est utilisée lorsqu'une seule apparition du motif est déjà un événement rare en soi. Un motif est considéré comme sur- ou sous- représenté si la valeur de la mesure choisie est au-delà ou en-deçà de seuils fixés de manière empirique. Dans ce contexte, le modèle exploratoire utilisé est markovien. Selon les hypothèses choisies, Obs(H) peut être compté de deux façons différentes. Dans le schéma recouvrant, si deux occurrences de H se recouvrent, elles sont comptées toutes les deux ; dans le schéma non recouvrant, seule l'une des deux est comptée. Ainsi, la séquence `TATCGTATATC` contient trois occurrences de `TAT` dans le premier cas, et deux seulement dans le second.

Le Z-score et la p-valeur peuvent être évalués expérimentalement, par génération aléatoire de séquences. D'autre part, des méthodes analytiques ont vu le jour, qui fournissent des solutions rigoureuses[8]. Dans certains cas, elles sont aussi plus rapides que la simulation. Deux principales approches analytiques sont utilisées actuellement. La première a été initiée par les travaux de Prum, Rodolphe, de Turkheim [142] et Schbath [159, 160]. Schématiquement, elle consiste à approcher la loi de $X_{\mathrm{H},n}$ par une loi normale ou une loi de Poisson, selon le comportement de $n\mathrm{Pr}(\mathrm{H})$ lorsque n tend vers l'infini (où $\mathrm{Pr}(\mathrm{H})$ désigne la probabilité d'observer H en une position quelconque). Une synthèse des méthodes de ce type est présentée en [148]. La deuxième approche fait appel à la combinatoire des mots et aux séries génératrices. La première étape est le calcul de la série génératrice bivariée

$$T(z,u) = \sum_{n\geq 0, r\geq 0} Pr(X_{\mathrm{H},n}=r) z^n u^r$$

dans laquelle le coefficient de $z^i u^j$ représente la probabilité qu'une séquence de longueur i contienne j fois le motif H. A partir de là, les calculs d'espérance et de variance se déduisent automatiquement. Nicodème, Salvy et Flajolet présentent une approche que je qualifierais de «généraliste» pour résoudre le problème [131]. Étant donné un motif et une chaîne de Markov, ils construisent un automate probabiliste dont la série génératrice associée, trouvée avec des méthodes standard de combinatoire, est le $T(z,u)$ cherché. Le point fort de cette méthode est sa généralité : elle s'applique à tout motif ou ensemble de motifs pouvant être décrit par une expression rationnelle. D'un autre côté, dans certains cas, le nombre d'états de l'automate est très élevé et, par conséquent son traitement très coûteux. L'approche de Mireille Régnier est différente tout en faisant appel aux mêmes concepts [145]. Lorsqu'on a affaire à un unique motif ou un ensemble fini de motifs, il n'est pas nécessaire de calculer explicitement l'automate. La méthode est fondée sur la décomposition du langage des séquences en une concaténation de langages associés aux motifs. Toutefois, la forme générale de la série génératrice ne dépend pas de ces motifs. On aboutit à des formules donnant directement les séries génératrices, et parfois leur coefficients, en fonction des motifs considérés.

Le calcul exact des probabilités d'occurrence est généralement coûteux en temps et peut donner lieu à des erreurs dues à l'instabilité numérique de certaines formules. D'autre part, les applications biologiques ne requièrent pas des valeurs rigoureusement exactes ; parfois un ordre de grandeur peut suffire. Toutes les méthodes ci-dessus donnent la possibilité de calculer asymptotiquement (lorsque n est grand) les valeurs d'intérêt. La sur-représentation d'un motif dans une séquence est, par définition, un événement rare. La théorie des grandes déviations peut donc s'appliquer dans ce cas. Plusieurs auteurs s'y sont intéressés : Régnier et Spankowski [147], Reinert, Schbath et Waterman [148] et Nuel [133] dont la récente thèse est consacrée à cette approche. J'ai récemment travaillé dans ce domaine avec Mireille Régnier [146]. Nos résultats dépendent, comme usuellement dans ce genre de problème, des *ensembles de corrélation* des motifs. L'ensemble de corrélation de deux motifs $\mathrm{H_i}$ et $\mathrm{H_j}$, noté $\mathcal{A}_{i,j}$, est l'ensemble des mots w tels qu'il existe un suffixe propre v de $\mathrm{H_i}$ qui satisfait $vw = \mathrm{H_j}$. Si $\mathrm{H_i} = \mathrm{H_j}$, alors il est appelé l'*ensemble d'auto-corrélation* de $\mathrm{H_i}$. Par exemple, l'ensemble de corrélation de $\mathrm{H_1} = \texttt{ATT}$ et $\mathrm{H_2} = \texttt{TAT}$ est $\mathcal{A}_{1,2} = \{AT\}$; l'ensemble d'autocorrélation de $\mathrm{H_1}$ est $\{\varepsilon\}$, tandis que celui de $\mathrm{H_2}$ est $\{\varepsilon,\texttt{AT}\}$. Le *polynôme de corrélation* de deux motifs $\mathrm{H_i}$ et

8. Elles s'appliquent lorsque la chaîne de Markov tend vers une distribution stationnaire.

$\mathrm{H_j}$ de longueurs respectives m_i et m_j est

$$A_{i,j}(z) = \sum_{w \in \mathcal{A}_{i,j}} \Pr(w) z^{|w|}.$$

Si $\mathrm{H_i} = \mathrm{H_j}$, alors on l'appelle le *polynôme d'autocorrélation* de $\mathrm{H_i}$. La *fonction de corrélation* est $D_{i,j}(z) = (1-z)A_{i,j}(z) + \Pr(\mathrm{H_j})z^{m_j}$. Lorsque $H_i = H_j$, on l'écrit simplement D_i.

Notre premier résultat, qui prolonge un théorème de [147], est une formule de grandes déviations pour la p-valeur d'un motif dans un modèle de Bernoulli et dans le schéma recouvrant.

Théorème 16 (Régnier, Denise 2001 [146]) *Soit* H_1 *un motif, et* r *son nombre d'occurrences observé dans une séquence de longueur* n*. Notons* $a = \frac{r}{n}$ *et supposons que* $a > \Pr(\mathrm{H}_1)$*. Alors*

$$\mathrm{pval}(\mathrm{H}_1) = \Pr(X_{\mathrm{H}_1,n} \geq r) \approx \frac{1}{\sigma_a \sqrt{n}} e^{-nI(a)}$$

où

$$\begin{aligned} I(a) &= a \ln\left(\frac{D_1(z_a)}{D_1(z_a) + z_a - 1}\right) + \ln z_a \ , \\ \sigma_a^2 &= a(a-1) - a^2 z_a \left(\frac{2D_1'(z_a)}{D_1(z_a)} - \frac{(1-z_a)D_1''(z_a)}{D_1(z_a) + (1-z_a)D_1'(z_a)}\right) \end{aligned}$$

et z_a *est la plus grande racine strictement inférieure à 1 de l'équation*

$$D_1(z)^2 - (1 + (a-1)z)D_1(z) - az(1-z)D_1'(z) = 0 \ .$$

Ce résultat est loin d'être aussi compliqué qu'il ne le paraît à première vue. Ce qui est important en pratique, c'est que le calcul de la p-valeur ne dépend que de la racine z_a de la dernière équation, dont le membre gauche est un polynôme dont le degré est de l'ordre de la longueur des motifs. Cela conduit à une approximation simple et rapide de la p-valeur.

En règle générale, l'approche analytique ne s'applique que dans des modèles homogènes[9]. La sur-représentation éventuelle d'un motif est évaluée indépendamment de tous les autres motifs qui peuvent apparaître dans la séquence. Cela peut faire apparaître ce qu'on appelle des *artefacts* : des motifs qui semblent sur-représentés parce que similaires à un motif réellement sur-représenté. Par exemple, dans le cas recouvrant, si `AATAAA` est très présent dans une séquence, les motifs `AAATAA` et `ATAAAA` ont de bonnes chances de l'être aussi. Le problème est donc, sachant qu'un motif est sur-représenté, de pouvoir en tenir compte pour extraire d'autres motifs éventuellement sur-représentés du «bruit» occasionné par les artefacts. Une variante de l'approche de [131] peut fournir une solution ; cependant, la taille de l'automate résultant de la contrainte supplémentaire peut se révéler encore une fois problématique. Avec les mêmes méthodes que précédemment, Mireille Régnier et moi avons obtenu le résultat suivant :

Théorème 17 (Régnier, Denise 2001 [146]) *Soient deux motifs* H_1 *et* H_2*. Supposons que le nombre* $\mathrm{Obs}(\mathrm{H}_1)$ *d'occurrences de* H_1 *dans une séquence réelle est égal à* r*, et que* $a = \frac{r}{n} \geq P(\mathrm{H}_1)$*. Alors l'espérance de* H_2 *conditionnée par les occurrences de* H_1 *s'écrit*

$$E(X_{\mathrm{H}_2}/\mathrm{Obs}(\mathrm{H}_1) = r) \sim n\alpha$$

9. On fait cependant allusion dans [130, 133, 148] à la possibilité de se placer dans des modèles de Markov cachés.

où α s'exprime en fonction de a, des fonctions d'autocorrélation et des probabilités des motifs :

$$\alpha = a\frac{D_{1,2}(z_a) \times D_{2,1}(z_a)}{D_1(z_a)(D_1(z_a) + z_a - 1)}$$

et z_a est calculé comme dans le théorème 16.

Dans un travail avec Mathias Vandenbogaert [65], nous avons commencé à valider ces résultats en les comparant à d'autres méthodes de calcul sur trois problèmes concrets d'analyse des génomes.

Ces travaux appellent des prolongements. En premier lieu, nos résultats dans le modèle de Bernoulli peuvent se généraliser au cas markovien, moyennant un peu plus de travail. Dans le théorème 17, nous pouvons ajouter que la variance est linéaire ; reste à la préciser. Le calcul de l'espérance conditionnée est un premier pas vers des solutions analytiques pour des modèles structurés. Une première étape de généralisation consisterait à conditionner par les occurrences d'un ensemble entier de motifs. On pourrait ainsi résoudre analytiquement les problèmes d'artefacts qui se posent, comme par exemple lors de la recherche de nouveaux signaux de polyadénylation dans [24]. Plus généralement, on peut envisager de généraliser les méthodes actuelles à des motifs plus complexes, comme les motifs structurés de [181]. Des pistes à ce propos peuvent être trouvées dans [84].

2.5.4 Modèles hétérogènes.

Modèles statistiques hétérogènes.

Comme les modèles homogènes, les modèles hétérogènes classiques sont fondés uniquement sur les distributions des oligonucléotides. Ils participent à la «génomique fondamentale» en ce sens qu'ils ne sont pas forcément appliqués à un ou des problèmes particuliers, mais peuvent aider à susciter de nouvelles hypothèses sur l'organisation et l'évolution des génomes. Dans [79], on représente une séquence par une suite de segments, chacun étant engendré selon une loi de Bernoulli particulière ; la loi de distribution des longueurs de segments est exponentielle. Dans [45], on représente les séquences par un modèle de Markov caché. Ce travail est pour beaucoup dans l'utilisation des chaînes de Markov pour les modèles prédictifs de recherche des gènes. C'est une belle illustration de l'importance des modèles exploratoires hétérogènes pour l'étude des génomes. Un modèle plus raffiné, appelé «*walking Markov model*», est présenté dans [82] : les probabilités de transition ne sont pas fixées *a priori*, mais varient selon un processus de Markov d'ordre 1.

Modèles structurés syntaxiques.

Le travail présenté en section 1.2.4 offre la possibilité d'engendrer des séquences obéissant à la fois à un modèle de Bernoulli et à une syntaxe, en utilisant une grammaire pondérée [66]. *A priori*, la généralisation à une linguistique markovienne ne pose pas de problème insurmontable. On peut construire une grammaire pondérée en effectuant le produit d'une grammaire algébrique et d'une chaîne de Markov, de façon similaire à ce qui est fait dans [131] dans le cas des automates pour l'approche analytique. (On se heurte toutefois à un problème récurrent : le nombre de poids à déterminer est exponentiel en fonction de l'ordre de la chaîne de Markov.)

Il existe par ailleurs des possibilités d'engendrer des séquences aléatoires selon des modèles syntaxiques. En effet, il n'est pas difficile d'utiliser une HMM «à l'envers» pour engendrer des

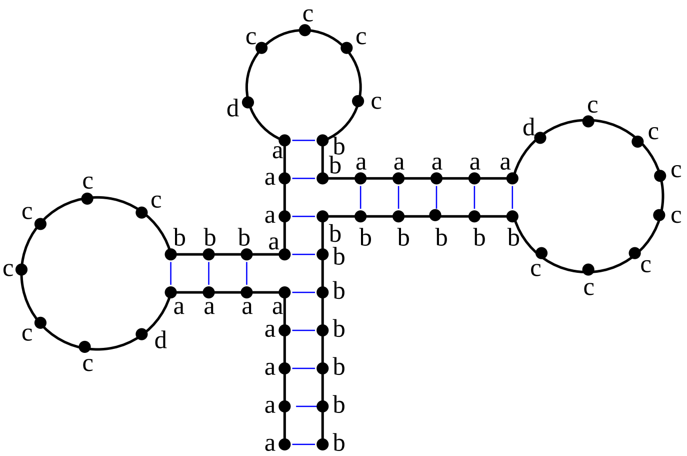


FIG. 2.7 – *Représentation de la structure secondaire qui correspond au mot* `aaaaaaadccccccbbbaaaadccccbbaaaaadcccccccbbbbbbbbbbb`.

séquences. Par exemple, le programme *HmmerEmit* de la boîte à outils d'analyse des génomes *GCG* construit des séquences qui respectent un profil HMM donné. L'apport majeur des grammaires pondérées est d'offrir une expressivité bien plus large que ces modèles classiques. En particulier, des interactions à longue distance peuvent être prises en compte. D'autre part, les grammaires pondérées peuvent inclure des paramètres qui ne sont pas nécessairement liés aux fréquences des nucléotides.

Voici un exemple simple de l'usage que l'on peut faire d'une grammaire pondérée. Il s'agit de modéliser des structures secondaires simplifiées, composées seulement de tiges (sans mésappariement) terminées par des boucles comportant au moins cinq nucléotides. Nous choisissons de coder chaque structure par un mot : les nucléotides en $5'$ et $3'$ d'un appariement sont représentés respectivement par les lettres a et b (qui peuvent être considérées comme des parenthèses ouvrante et fermante) ; chaque nucléotide à l'intérieur d'une boucle est représenté par la lettre c, sauf le premier à qui on attribue la lettre d. La figure 2.7 présente un mot et la structure qui lui correspond. Ce codage est une variante de la représentation des structures secondaires par des mots du langage de Motzkin [164, 182]. Le langage des structures secondaires ainsi définies est algébrique ; en voici une grammaire :

$$\begin{array}{lcl} S & \to & \mathtt{a}\,S\,\mathtt{b}\,S \mid \mathtt{a}\,S\,\mathtt{b} \mid \mathtt{d}\,T\,\mathtt{c} \\ T & \to & \mathtt{c}\,T \mid \mathtt{c\ c\ c} \end{array}$$

Le rôle de la lettre `d` est de permettre de faire varier le nombre moyen de tiges-boucles des structures aléatoires en jouant sur la pondération. De même, la longueur moyenne des boucles peut être modulée en modifiant le poids de la lettre `c`. Supposons par exemple que l'on veuille engendrer des structures de longueur 60 nucléotides. En calculant la série génératrice et appliquant la formule de calcul des $\mu_i(\pi)$ de la page 17, on montre aisément que, si tous les poids valent 1 (c'est-à-dire si la génération est uniforme), alors le nombre moyen de boucles est environ égal à 5,62, et chaque boucle comporte 6,68 nucléotides en moyenne. En posant $\pi(\mathtt{a}) = 16$, $\pi(\mathtt{c}) = 2$ et $\pi(\mathtt{b}) = \pi(\mathtt{d}) = 1$, on obtient des structures ayant en moyenne trois boucles de longueur 5,80 nucléotides. Si l'on veut compter *exactement* trois boucles dans chaque structure engendrée, on peut concevoir une grammaire différente :

$$S \;\to\; \mathtt{a}\,S\,\mathtt{b} \mid T\,T\,T \mid T\,U \mid U\,T$$

$$
\begin{array}{rcl}
T & \to & \mathtt{a}\,T\,\mathtt{b} \mid \mathtt{a}\,V\,\mathtt{b} \\
U & \to & \mathtt{a}\,U\,\mathtt{b} \mid \mathtt{a}\,T\,T\,\mathtt{b} \\
V & \to & \mathtt{c}\,V \mid \mathtt{c\,c\,c\,c\,c}
\end{array}
$$

Ici, il est inutile d'ajouter une lettre `d` : le nombre de boucles est fixé par la grammaire. Le poids de `c` détermine la longueur moyenne des boucles. Ces grammaires peuvent être complexifiées pour modéliser des structures de façon plus «réaliste». Ainsi, par exemple, on peut modéliser des mésappariements dans les tiges en ajoutant la règle $T \to \mathtt{x}\,T\,\mathtt{x}$ à la grammaire ci-dessus.

Un logiciel de génération de séquences, baptisé GenRGenS (Generation of Random Genomic Sequences) est en cours de développement grâce au travail initial de Frédéric Sarron [156] et à celui de Yann Ponty qui a pris la relève [139]. Notre projet est d'en faire un outil capable de prendre en compte aussi bien les modèles simples que les modèles hétérogènes les plus complexes. Les quelques outils qui existent à ce jour ne gèrent que des modèles simples [10]. La version actuelle de GenRGenS prend en compte le modèle markovien (avec possibilité de tenir compte d'une phase quelconque), certaines chaînes de Markov cachées et les grammaires pondérées. Les modèles sont décrits par des fichiers. Voici par exemple un fichier de description pour la première grammaire pondérée définie ci-dessus.

```
TYPE = GRAMMAR
SYMBOLS = LETTERS
RULES
  S -> a S b S;
  S -> a S b;
  S -> d T c;
  T -> c T;
  T -> c c c;
WEIGHTS =
  a 16
  c 2
```

Le fichier suivant décrit un modèle markovien d'ordre un à trois phases. Les nombres qui suivent les dinucléotides représentent les fréquences observées sur la séquence de référence dans les phases 0, 1 et 2.

```
TYPE = MARKOV
ORDER = 1
PHASE = 3
SYMBOLS = LETTERS
FREQUENCIES =
  aa  97 112  71   ac  67  59  69   ag   45  63  75   at  82 103  51
  ca  76  59  93   cc  47 103  65   cg   83  80 134   ct  94  62  58
  ga 124  57  61   gc 132 109 101   gg  117  54 109   gt  90  78  50
  ta  40  38  65   tc  58  79  66   tg   53 124 145   tt  78 103  70
```

Les chaînes de Markov cachées peuvent être gérées grâce à un mécanisme d'*alias* qui permet de définir une chaîne sur un alphabet étendu, puis de substituer certaines lettres à d'autres en

10. La boite à outils d'analyse des séquences GCG contient trois utilitaires à ce propos : le premier (Shuffle) «mélange» une séquence en conservant sa composition en nucléotides ; le second (HmmerEmit) engendre des séquences qui respecte un profil HMM donné ; le troisième (Corrupt) introduit des mutations aléatoires dans une séquence donnée. De son côté, Coward [51] a implémenté et mis en ligne l'algorithme de génération en fréquences exactes d'oligonucléotides de [113].

fin de traitement. Une particularité importante (en cours d'implémentation) du programme est la possibilité d'engendrer des séquences composites, c'est-à-dire dont les différentes zones obéissent à des modèles de natures très différentes. Voici par exemple un modèle grammatical très fruste d'un chromosome d'eucaryote inférieur.

```
TYPE = GRAMMAR
SYMBOLS = WORDS
RULES
  Chromosome -> Telomere5 Reste Centromere Reste Telomere3;
  Reste -> Intergenique ZonePromotrice Gene ZoneTerminatrice Reste;
  Reste -> ;
  Gene -> GeneProteique;
  Gene -> tRNA;
  Gene -> rRNA;
  Gene -> snRNA;
```

Chaque symbole terminal (`Telomere5`, `Centromere`, `Intergenique`, `ZonePromotrice`, `Gene-Proteique`, etc.) de cette grammaire représente un type de zone. On peut s'arrêter là (et engendrer des mots sur l'alphabet des symboles terminaux de cette grammaire), ou bien aller plus loin en associant à chaque type de zone un modèle particulier (par exemple markovien pour les zones intergéniques, grammatical pour les tRNA). Le résultat de la génération sera une séquence sur l'alphabet des nucléotides.

La version actuelle du logiciel est utilisée dans les travaux présentés en section 2.6. De plus, dans le cadre d'une collaboration avec le groupe de Serge Dulucq (LaBRI, Bordeaux), nous allons appliquer la génération de structures secondaires pour évaluer et étalonner des algorithmes d'alignement de structures secondaires [75]. Il s'agit de comparer les scores d'alignement de structures réelles à ceux de structures aléatoires ayant des propriétés similaires. Un autre projet est en discussion avec l'équipe de Jacques Nicolas à Rennes. Il consiste à utiliser la génération aléatoire pour tester des algorithmes d'inférence grammaticale à partir de séquences génomiques.

D'autres types de modélisation vont être intégrées dans le logiciel : motifs PROSITE, profils et profils généralisés... D'un point de vue plus théorique, nous allons étudier la possibilité d'élargir la faculté d'expression des modèles, par exemple en utilisant le formalisme des *string variable grammars* [162]. Le problème de la génération aléatoire de mots selon des grammaires non algébriques, n'a, à ma connaissance, jamais été étudié.

Contraintes de motifs dans le modèle exact.

En collaboration avec Dominique Barth et Romain Rivière, j'étudie la possibilité d'ajouter des paramètres syntaxiques au modèle linguistique exact. Le mémoire de DEA de Romain Rivière présente les résultats de nos premières investigations [151]. Il s'agit, dans un premier temps, d'imposer des contraintes sur le nombre d'occurrences de certains motifs. On se donne les nombres d'occurrences d'oligonucléotides de la séquence de référence et un ensemble de motifs, chacun étant associé à un nombre exact, minimal ou maximal d'occurrences. Par exemple, il peut s'agir d'engendrer une séquence de même composition en trinucléotides que `ACTACTGTAAACTAC` contenant exactement une fois le motif `CTACTA` et au moins une fois le motif `TGTA`. Bien entendu, toutes les séquences possibles doivent être équiprobables. Ce problème a

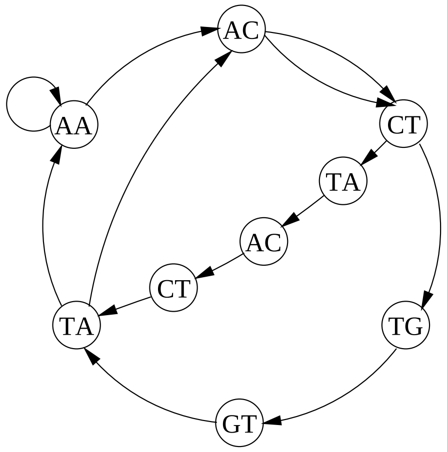


FIG. 2.8 – *Multigraphe pour $l = 3$ et la séquence de référence* `ACTACTGTAAACTAC` *(cf. figure 2.6(b)), avec la contrainte d'obtenir au moins une fois chacun des motifs* `CTACTA` *et* `TGTA`*.*

été étudié par Trifonov [176], mais sa solution ne garantit pas l'uniformité de la génération. Notre approche s'inspire du cas non contraint et y ajoute un zeste de génération par rejet. La première étape consiste à construire un multigraphe tenant compte de toutes les contraintes «au moins» et transformant les contraintes «exactement» en contraintes «au moins». Dans notre exemple, on construira[11] un multigraphe représentant toutes les séquences de composition en trinucléotides adéquate qui contiennent au moins une fois chacun des motifs `CTACTA` et `TGTA`. L'algorithme consiste, partant du multigraphe sans contraintes, à dupliquer certains sommets pour construire des passages obligés aux chemins eulériens (le nombre d'arcs reste constant). La figure 2.8 présente le multigraphe obtenu pour notre exemple. (On remarque en passant que la seconde contrainte était déjà respectée dans le multigraphe d'origine.) Le multigraphe une fois déterminé, le procédé de génération est identique au cas sans contrainte, à ceci près qu'une étape de rejet est effectuée pour ne conserver que les chemins respectant les contraintes «exactement» et «au plus».

La complexité moyenne du pseudo-algorithme peut être évaluée. On sait en effet calculer le nombre de circuits eulériens[12] dans un multigraphe eulérien G [88] :

$$N(G) = \mathcal{T}(G) \prod_{v \in V(G)} (\deg^+(v) - 1)! \,,$$

où $V(G)$ est l'ensemble des sommets de G, $\deg^+(v)$ désigne le degré sortant du sommet v (le nombre d'arcs dont v est origine) et $\mathcal{T}(G)$ est le nombre d'anti-arbres couvrants de G enracinés en un sommet fixé quelconque. Cette formule découle directement de la relation évoquée en section 2.5.2, page 42. La *matrice de Kirschoff* de G est égale à $D - A$, où A est la matrice d'incidence de G, et D est une matrice diagonale dont chaque coefficient $D_{i,i}$ est le degré sortant du sommet v_i. Le nombre $\mathcal{T}(G)$ est égal au cofacteur du terme en dernière ligne et dernière colonne de la matrice de Kirschoff. Ces considérations permettent

11. Certaines contraintes peuvent être incompatibles entre elles ou avec la composition linguistique imposée. dans ce cas l'algorithme renvoie un multigraphe vide.

12. On considère que G est circuit-eulérien. Au besoin, on lui ajoute un arc ; cela n'a pas d'incidence sur le traitement.

de calculer la probabilité qu'une séquence contienne au moins r fois un certain motif. Des calculs élémentaires permettent d'en déduire la probabilité exacte de rejet [13].

Notre méthode est en cours d'expérimentation. La construction du graphe ne pose pas de problème ; elle est linéaire en la somme des tailles de la séquence et des motifs. Le rejet pourrait rendre le pseudo-algorithme inefficace dans certains cas. Il nous faut étudier sa complexité dans un cadre général en fonction des différents types de contraintes. D'autres méthodes de génération sont aussi à l'étude. Nous envisageons d'enrichir les contraintes, par exemple en imposant un ordre d'apparition, ou en imposant une position déterminée à certains motifs.

2.6 Deux études en bioinformatique des génomes.

Cette section est consacrée à deux études en cours. Les modèles exploratoires y jouent un rôle. Cependant, il ne s'agit pas seulement –loin de là– d'illustrer l'application de ces modèles en analyse des génomes. Ces travaux sont éminemment pluridisciplinaires : ils nécessitent des compétences en biologie moléculaire, en bioinformatique des génomes et dans plusieurs domaines de l'informatique. Ils font appel à la modélisation bioinformatique, à l'expérimentation biologique et à des développements informatiques, tant théoriques que pratiques.

2.6.1 Étude des zones en $5'$ des gènes protéiques de levure.

Il s'agit de travaux effectués par David Abergel, Rachid Gherbi, Michel Termier et moi-même. Le but de cette étude est de concevoir des modèles de zones promotrices de gènes, en alliant l'approche structurelle tridimensionnelle et l'approche syntaxique. Un travail exploratoire a été effectué par David Abergel durant son stage de DEA [3] sous la direction de Michel Termier et Rachid Gherbi. Ont été sélectionnés, dans le génome de *S. cerevisiae*, les 3102 gènes protéiques connus qui possèdent une zone intergénique de longueur au moins 350 nucléotides en amont du codon START. L'étude de la composition linguistique de ces zones montre clairement que la portion de longueur environ 50 nucléotides située immédiatement en amont du codon START se distingue du reste de l'intergénique. L'autre volet du travail de David Abergel a consisté à étudier la courbure de l'ADN dans la zone entourant le codon START, selon le modèle de Bolshoy et al. [32], implémenté au LIMSI [90, 91]. Chacune des séquences a été modélisée (modèle descriptif) par la suite des *abscisses curvilignes* de ses nucléotides (à chaque position de nucléotide est associé un nombre qui représente d'une certaine façon la courbure autour de cette position). L'ensemble entier des séquences a été modélisé par une unique courbe représentant, pour chaque position de nucléotide, l'abscisse curviligne moyenne des 3102 séquences en cette position. On a ainsi pu observer une différence sensible entre la courbure moyenne avant et après le codon START, comme cela avait été remarqué chez un certain nombre de procaryotes dans [33]. Toutefois, cette différence se présente aussi si l'on remplace les 3102 séquences par des séquences aléatoires obéissant à un modèle markovien hétérogène qui prend en compte les compositions en oligonucléotides de l'intergénique et du codant. La même expérience a été effectuée en s'intéressant au +1 de transcription au lieu du codon START. Cependant, comme peu d'informations sont disponibles sur les +1 de transcription des gènes de *S. cerevisiae*, seules 46 séquences pour lesquelles leur localisation est

13. C'est un calcul similaire qui a donné naissance à l'approche analytique de [142] (voir section 2.5.3). L'évaluation exacte des probabilités n'est envisageable que pour des petites séquences, car, pour des séquences longues, la taille du graphe (et donc du déterminant) peut augmenter exponentiellement en l'ordre de la chaîne de Markov. C'est pourquoi, dans [142], on fait usage d'expressions asymptotiques.

connue sans ambiguïté ont pu être sélectionnées. Un maximum de courbure moyenne est observé à proximité du +1 de transcription. Il ne semble pas être lié à une modification de la composition nucléotidique à cet endroit ; mais les investigations doivent être approfondies et les résultats confirmés sur un plus grand nombre de données.

Ici, l'utilisation d'un modèle hétérogène a permis de mettre en évidence une corrélation entre linguistique et courbure chez *S. cerevisiae* (il serait intéressant de revisiter le travail de [33] sous cet angle). S'il se confirme qu'un événement de courbure particulier apparaît à proximité du +1 de transcription sans être lié à une conservation de la séquence primaire, il faudra vérifier expérimentalement qu'il joue un rôle dans le démarrage de la transcription. Pour cela, nous projetons d'engendrer *in silico* d'une part des sites ne possédant pas cet événement de courbure mais présentant une séquence primaire similaire aux sites observés sur la levure ; d'autre part des sites possédant cet événement de courbure mais présentant une séquence primaire très différente des sites de levure. L'expérimentation *in vivo* de ces sites permettra de déterminer l'importance relative de la séquence primaire et de la courbure dans l'initiation de la transcription. Plus généralement, nous envisageons d'étudier d'un point de vue théorique les rapports entre courbure, linguistique et syntaxe. À composition en oligonucléotides fixée, quel est l'éventail des courbures que l'on peut obtenir ? Peut-on déterminer l'ensemble des séquences qui présentent un événement de courbure donné ? Ici encore, la génération de séquences aléatoires nous sera utile.

2.6.2 Modélisation des sites de décalage de cadre de -1 en traduction.

Ce travail est effectué dans une collaboration entre l'équipe de Jean-Pierre Rousset à l'IGM (Génétique Moléculaire de la Traduction), celle de Michel Termier (Bioinformatique des Génomes), Jean-Paul Forest et Christine Froidevaux de l'équipe Intelligence Artificielle et Systèmes d'Inférences du LRI, et moi-même. Le travail a commencé dans le cadre du stage de DEA de Jean-Paul Forest [89]. Notre démarche et nos premiers résultats ont été présentés dans [62]. Des résultats expérimentaux plus récents sont présentés dans le rapport de stage de DEA de Michaël Bekaert [26].

Les mécanismes de base de la traduction, tels que donnés en section 2.2.1, sont universels. Cependant, certains ARN messagers spécifiques donnent lieu à des déviations du schéma général, collectivement désignés sous le nom de *recodage* [168]. Le décalage de cadre de lecture (*frameshift*) en -1 est l'un de ces mécanismes de décodage non conventionnels de l'information génétique. Lors de la traduction, à un endroit particulier de l'ARN messager, le ribosome se décale d'un nucléotide en amont puis reprend la lecture des codons dans la nouvelle phase (cadre de lecture) dans laquelle il se trouve. Ainsi, à partir d'un même ARN messager, deux protéines différentes peuvent être engendrées, selon que le décalage se produit ou pas. Ce mécanisme constitue un processus normal qui est responsable d'un contrôle probabiliste très fin de l'expression : les deux produits, qui possèdent en général des fonctions différentes, sont synthétisés dans des proportions déterminées. De tels phénomènes sont observés principalement dans des génomes simples tels que des virus ou des transposons. De très rares exemples de décalage en -1 sont connus chez les procaryotes, et aucun chez les eucaryotes. Cependant, le fait que la machinerie de traduction sait interpréter les signaux de décalage viraux permet d'imaginer que des gènes non viraux emploient des mécanismes semblables. Les gènes présentant un événement de décalage de cadre ont un message génétique qui, par définition, n'est pas en phase. Ils risquent donc d'être répertoriés comme séquences non codantes ou pseudogènes. Il est donc crucial de disposer d'outils fiables de prédiction *in silico* de ces gènes.

Chez les virus d'eucaryotes, le signal responsable du décalage de cadre en -1 contient un heptanucléotide (le *motif glissant*) et une structure secondaire particulière située quelques nucléotides en aval de celui-ci. Le motif glissant a pour forme canonique $X\ XXY\ YYZ$, avec $Y =$ A ou U et $Z \neq$ G (les espaces marquent le cadre de lecture). Bien sûr, il existe des variantes à cette forme canonique. La structure secondaire est soit un pseudo-nœud, soit une simple tige-boucle. L'observation des gènes à décalage de cadre connus et les mutations effectuées pour leur étude, notamment dans l'équipe de Brierley [38, 37, 127], ont permis de définir des paramètres d'importance (distance entre le motif glissant et la structure secondaire, longueur et stabilité de la tige,...) pour que le décalage s'effectue à un taux non négligeable. Deux tentatives de conception de modèles prédictifs ont été effectuées [104, 120]. Ces modèles n'ont pas permis pour l'instant de trouver de nouveaux sites authentiques de décalage. Cela tient principalement à leur très faible spécificité.

Nous travaillons à la conception d'un modèle prédictif plus fin. Pour cela, nous nous appuyons sur les données connues, rassemblées dans une banque de séquences qui contient des exemples positifs (séquences sauvages ou mutantes qui sont sujettes au décalage) et des exemples négatifs (mutants qui n'obéissent pas au phénomène). Les propriétés que nous jugeons *a priori* susceptibles d'être pertinentes dans le phénomène sont intégrées dans une représentation des données adéquate. Puis, à l'aide d'un logiciel d'apprentissage de concepts nous dégageons des règles permettant de discriminer les exemples positifs des négatifs. L'analyse de ces règles débouche sur un modèle exploratoire des sites de décalage de cadre. L'étape suivante consiste à constituer un nouvel ensemble de sites qui obéissent au modèle, ou bien qui n'y obéissent pas mais en sont proches. Ces sites sont soit totalement artificiels, soit détectés dans des bases de données génomiques (sans être connus pour être soumis au décalage de cadre). Leur comportement quant au décalage est évalué *in vivo* : les séquences à tester sont clonées dans un vecteur d'expression qui permet de mesurer le taux de décalage [167]. Puis le modèle est évalué en comparant les prédictions théoriques et les résultats expérimentaux. Les nouveaux sites sont intégrés dans la banque de séquences, et un nouveau cycle de modélisation / construction de sites / validation expérimentale peut commencer. Lorsque le modèle sera jugé suffisamment fiable, il pourra être associé à une algorithmique efficace pour constituer un modèle prédictif.

Un autre but de notre travail est de prendre en compte l'aspect quantitatif du décalage de cadre. Nous savons qu'il se produit à un taux donné, correspondant aux proportions des deux protéines produites. Or, dans tous les modèles existants, il est considéré comme un mécanisme en tout ou rien (en fait nous considérons qu'un exemple est négatif si le taux est $\leq 2\%$, positif s'il est $\geq 5\%$). Nous projetons d'intégrer dans la modélisation des éléments permettant de rendre compte du taux de décalage plus finement que par le simple dépassement d'un seuil.

Nous avons la chance d'être confrontés à un problème particulièrement bien spécifié : les acteurs du phénomène sont peu nombreux et bien identifiés ; le schéma général du modèle (site glissant et structure secondaire) est appuyé par une interprétation biologique convaincante ; les expérimentations effectuées sur des sites sauvages et mutants fournissent des données quantitatives fiables. C'est donc un problème pour lequel la démarche de modélisation ne peut être que fructueuse. En contrepartie, nous disposons de peu de données alors que les paramètres à prendre en compte sont nombreux. En effet, il y a actuellement environ 250 séquences dans la base, chacune d'entre elles étant décrite par 120 attributs. Tous ne sont probablement pas pertinents, mais on ne peut pas en décider *a priori* ; et il existe certainement des paramètres importants que nous n'avons pas encore pris en compte. De plus, l'ensemble des données n'est pas homogène : beaucoup de mutants concernent une même partie très spécifique du site pour un virus particulier.

Les premiers résultats de l'étape d'apprentissage ont permis de dégager quelques règles qui couvrent beaucoup des exemples considérés. Toutefois, le travail n'est pas encore assez avancé pour que nous disposions d'un modèle intégral du phénomène. Pour l'instant, l'évaluation expérimentale s'effectue sur des sites construits en choisissant de façon déterministe des mutations *ad hoc* sur des séquences connues. Ultérieurement, nous ferons appel à la recherche dans les banques génomiques et à la génération aléatoire de séquences selon les critères syntaxiques et statistiques du modèle que nous aurons à évaluer. En ce qui concerne l'aspect quantitatif du phénomène, nous avons mis en évidence l'importance d'un paramètre négligé jusqu'à présent : la structure primaire de l'*espaceur* (*spacer*), zone située entre le motif glissant et la structure secondaire, a une forte influence sur le taux de décalage.

Pour progresser, nous avons besoin de déterminer plus précisément un ensemble de paramètres pertinents. Il nous faudra réfléchir sur une représentation des données permettant notamment de prendre en compte plus finement les spécificités de la structure secondaire et les variations du taux de décalage. Nous envisageons d'intégrer la notion de séquentialité dans l'apprentissage des concepts, car le caractère séquentiel de la traduction est primordial dans le phénomène de décalage. Cela pourra conduire à concevoir et réaliser des variantes des outils traditionnels d'apprentissage de concepts. En ce qui concerne la prédiction, nous pourrons être amenés à adapter à notre problématique des algorithmes classiques de prédiction de structures secondaires. En effet, le fait de savoir quel type de structure secondaire nous recherchons dans les sites de frameshift (tige ou pseudo-nœud possédant certaines propriétés additionnelles) peut contribuer à obtenir des algorithmes plus précis et plus rapides que les algorithmes généraux. Enfin, notre méthode de validation par création *in silico* de sites de frameshift artificiels pose le problème de la génération aléatoire de structures secondaires qui dépassent le cadre algébrique de mes travaux sur la génération de séquences aléatoires. Toutes ces perspectives posent des problèmes qui demandent des compétences variées, et que seul un groupe pluridisciplinaire peut résoudre. A plus long terme, le même type d'approche pourrait être mis en œuvre pour la recherche d'autres événements de traduction non conventionnels pour lesquels les biologistes du groupe ont pu dégager un certain nombre de données expérimentales.

Perspectives.

Un certain nombre de prolongements possibles des travaux en cours ont été présentés tout au long du mémoire. Je voudrais ici suggérer quelques questions d'ordre plus général et présenter brièvement la façon dont j'aimerais voir évoluer mes recherches à moyen terme.

Il reste énormément de travail à faire du côté informatique pour améliorer aussi bien la précision que l'expressivité des modèles exploratoires. Par exemple, le calcul des poids dans les grammaires algébriques est un problème qui n'est pas résolu dans son entier. Dans ce même cadre des grammaires pondérées, des formalismes plus expressifs, comme par exemple les *string variable grammars*, doivent être étudiés. En ce qui concerne l'approche analytique, il faudra tenter d'intégrer des paramètres plus contraignants, notamment en conditionnant le calcul des probabilités à la sur-représentation de plusieurs motifs.

D'un autre côté, les avancées théoriques (du côté informatique) devront être confrontées à la réalité de la bioinformatique. Les travaux que nous effectuons à Orsay sont, de ce point de vue, un excellent «banc d'essai» pour les modèles exploratoires structurés. Il sera important de les valider dans d'autres contextes. Les collaborations prévues avec Serge Dulucq (comparaison de structures secondaires) et Jacques Nicolas (inférence grammaticale) vont dans ce sens. Les modèles structurés peuvent certainement être appliqués dans d'autres champs que celui de l'analyse des séquences. Je pense en particulier à la phylogénie, pour laquelle l'étude des arbres aléatoires est d'une grande importance. Pour une utilisation et une validation plus larges des modèles structurés, il nous appartient de faire en sorte qu'un outil fiable et convivial de modélisation et de génération soit bientôt mis à la disposition de la communauté bioinformatique.

L'utilisation des modèles exploratoires soulève aussi bien des problèmes théoriques qui ne sont pas du ressort de l'informatique. Une question importante est : quels critères prendre en compte pour la conception d'un modèle exploratoire ? Le problème se pose déjà dans le modèle markovien homogène. À partir d'une ou plusieurs séquences de référence, il existe des procédés objectifs pour déterminer un ordre adéquat pour la chaîne de Markov. Cependant, il est pertinent de s'interroger, comme le fait Marie-France Sagot [154] dans le cadre de l'identification des promoteurs, sur le choix du contexte (donc des séquences de référence elles-mêmes) à utiliser pour juger de la sur-représentation d'un motif. La question est sans doute plus délicate encore pour les modèles structurés : en sus du contexte, il faut tenter de déterminer l'ensemble des paramètres les plus pertinents parmi tous ceux qui participent à la structure.

Même s'ils sont importants, il ne faut pas perdre de vue que les modèles exploratoires ne sont qu'un moyen parmi d'autres pour compléter et affiner nos connaissances de l'information génomique. On ne peut véritablement prétendre avoir compris un phénomène que si on sait en concevoir un modèle prédictif fiable qui peut donner lieu à une interprétation biologique.

On ne peut raisonnablement appliquer ce modèle que s'il est assorti d'une algorithmique efficace permettant de travailler à l'échelle des génomes. À mon sens, ces buts ne peuvent être atteints qu'en travaillant au sein de groupes soudés à multiples compétences, dans lesquels chacun a fait l'effort de s'initier aux disciplines des autres. (C'est vrai aussi, bien sûr, pour d'autres champs de recherches, comme je l'ai appris au cours d'autres travaux collaboratifs, dont en particulier le test de logiciel.) Dans le cadre de nos recherches, les aller-retour entre modélisation formelle, expérimentation biologique et développement logiciel me semblent être une démarche très prometteuse. Le travail de bioinformatique ne consiste pas seulement à analyser les données du séquençage ; il conduit à formuler des hypothèses qui ouvrent la voie à de nouvelles expérimentations biologiques qui, à leur tour, conduisent à l'affinement des modèles. Un de mes vœux les plus chers est de pouvoir continuer à travailler dans cet esprit.

Annexe : résumés des travaux en combinatoire énumérative.

Deux paramètres sur les mots de Dyck.

Travail effectué en collaboration avec Rodica Simion [67]. Nous avons défini et étudié deux paramètres sur les chemins de Dyck : le *poids en pyramides* et le *nombre de paires extérieures*. Nous avons établi des formules closes pour les fonctions génératrices respectives des chemins de Dyck selon l'un et l'autre de ces paramètres. Les résultats ont été obtenus en enchaînant plusieurs méthodes : la méthodologie de Schützenberger, les q-grammaires et des interprétations combinatoires à base de partitions non croisées.

Dénombrement d'animaux dirigés pour des familles infinies de réseaux.

Travail effectué en collaboration avec Sylvie Corteel et Dominique Gouyou-Beauchamps [49]. Dans [34], Bousquet-Mélou et Conway ont étudié les animaux dirigés sur une famille infinie de réseaux bidimensionnels réguliers non planaires. En utilisant des équivalences avec des modèles de particules dures, ils ont obtenu des équations algébriques satisfaites par les fonctions génératrices selon l'aire de ces animaux. En utilisant une généralisation des «empilements de structures» de Viennot, nous avons fourni des preuves bijectives de leurs résultats. Cette approche nous a permis d'aboutir à des formules de dénombrement qu'on ne pouvait pas obtenir à partir des équations algébriques. Nous avons étendu nos résultats à une autre famille infinie de réseaux.

Nature des séries génératrices de systèmes ECO.

Travail effectué en collaboration avec Cyril Banderier, Mireille Bousquet-Mélou, Philippe Flajolet, Danièle Gardy et Dominique Gouyou-Beauchamps [18]. Un système ECO [19] est une sorte de spécification d'objets combinatoires. Elle consiste en un ensemble de règles qui représente un arbre infini. Chaque nœud de l'arbre correspond à un objet combinatoire ; chaque chemin de la racine à un nœud décrit le processus de génération de l'objet. Nous avons étudié les liens entre les propriétés structurelles de tels arbres, ou plutôt des systèmes de règles associés, et la nature (rationnelle, algébrique ou transcendante) de la série génératrice qui leur correspond. Cette série énumère les nœuds de niveau donné de l'arbre, *i.e.*, les objets de taille donnée. Nous avons ainsi dégagé un certain nombre de conditions suffisantes pour prédire la nature de la série génératrice directement à partir du système de règles.

Coloriage de chemins de permutations dans les arbres.

Travail effectué en collaboration avec Dominique Barth, Sylvie Corteel, Danièle Gardy et Mario Valencia-Pabon [50]. Le problème du coloriage de chemins de permutations sur un graphe G à n sommets $\{1,2,\ldots,n\}$ est le suivant : étant donné une permutation σ de $\{1,2,\ldots,n\}$, quel est le nombre de couleurs nécessaires pour dessiner tous les chemins menant de chaque sommet i à $\sigma(i)$ de façon que deux chemins de même couleur ne passent pas par une même arête? Ce problème est lié à l'étude des communications dans les réseaux à fibres optiques (chaque couleur représente une longueur d'onde). Nous avons démontré qu'il est NP-difficile sur les arbres binaires ainsi que sur les arbres ayant seulement deux sommets de degré supérieur à deux. Le résultat reste vrai si les permutations sont des involutions. Nous avons aussi établi que le même problème appliqué aux permutations circulaires sur les arbres de degré maximum supérieur ou égal a 4 est aussi NP-complet. Nous avons donné un minorant du nombre de couleurs nécessaire sur un réseau arbitraire. Nous avons démontré que, sur un réseau linéaire, le nombre moyen de couleurs nécessaires est $n/4 + o(n)$. Nous avons étendu ce résultat pour obtenir un majorant du nombre moyen de couleurs nécessaire sur des arbres généraux.

Bibliographie

[1] IEEE standard for binary floating-point arithmetic. Tech. Rep. ANSI-IEEE Standard 754–1985, American National Standards Institute, New York, 1985.

[2] *Larousse encyclopédique universel*. Larousse-Bordas, 1997.

[3] D. Abergel. Caractérisation syntaxique et spatiale des régions intergéniques chez *S. cerevisiae*. Rapport de stage de DEA "Approches structurales, fonctionnelles et évolutives des génomes", université Paris-Sud XI, juin 2001.

[4] D.J. Aldous. A random walk construction of uniform spanning trees and uniform labelled trees. *SIAM Journal on Discrete Mathematics*, 3(4):450–465, 1990.

[5] L. Alonso. Uniform generation of a Motzkin word. *Theoretical Computer Science*, 134:529–536, 1994.

[6] L. Alonso and R. Schott. *Random generation of trees*. Kluwer academic publishers, 1995.

[7] S.F. Altschul and B.W. Erickson. Signifiance of nucleotide sequence alignments: a methode for random sequence permutation that preserves dinucleotide and codon usage. *Mol. Biol. Evol.*, 2(6):256–538, 1985.

[8] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[9] The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 408:796–815, 2000.

[10] D.B. Arnold and M.R. Sleep. Uniform random generation of balanced parenthesis strings. *ACM Trans. Programming Languages and Systems*, 2(1):122–128, 1980.

[11] D.G. Arquès, J.C. Lapayre, and C.J. Michel. Identification and simulation of shifted periodicities common to protein coding genes of eukaryotes, prokaryotes and viruses. *Journal of Theoretical Biology*, 172:279–291, 1995.

[12] D.G. Arquès and C.J. Michel. Periodicities in introns. *Nucleic Acids Research*, 15(18):7581–7592, 1987.

[13] D.G. Arquès, C.J. Michel, and K. Orieux. Identification and simulation of new non-random statistical properties common to different populations of eukariotic coding genes. *Journal of Theoretical Biology*, 161:329–342, 1993.

[14] M. Ashburner. A biologist's view of the *Drosophila* genome annotation assessment project. *Genome research*, 10:391–393, 2000.

[15] M. D. Atkinson and J. R. Sack. Generating binary trees at random. *Information Processing Letters*, 41:21–23, 1992.

[16] S. Audic and J.M. Claverie. Self-identification of protein-coding regions in microbial genomes. *Proceedings of the National Academy of Science USA*, 95(17):10026–10031, 1998.

[17] P. Baldi, S. Brunak, Y. Chauvin, J. Engelbrecht, and A. Krogh. Periodic sequence patterns in human exons. In *ISMB'95*, pages 30–38. The AAI Press, 1995.

[18] C. Banderier, M. Bousquet-Mélou, A. Denise, Ph. Flajolet, D. Gardy, and D. Gouyou-Beauchamps. Generating functions for generating trees. *Discrete Mathematics*, 2001. 26 pages. A paraître.

[19] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. ECO: a methodology for the enumeration of combinatorial objects. *Difference Equations and Applications*, 5:435–490, 1999.

[20] E. Barcucci, A. Del Lungo, and E. Pergola. Random generation of trees and other combinatorial objects. *Theoretical Computer Science*, 218(2):219–232, 1999.

[21] E. Barcucci, R. Pinzani, and R. Sprugnoli. Génération aléatoire des animaux dirigés. In *Actes de l'Atelier de Combinatoire franco-québecois*, volume 10 of *Publications du LACIM*, pages 17–25. Université du Québec à Montréal, 1991.

[22] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of underdiagonal walks. In Pierre Leroux and Christophe Reutenauer, editors, *Actes de FPSAC'92*, volume 11 of *Publications du LACIM*. Université du Québec à Montréal, 1992.

[23] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of directed animals. *Theoretical Computer Science*, 127(2):333–350, 1994.

[24] E. Beaudoing, S. Freier, J. Wyatt, J.M. Claverie, and D.Gautheret. Patterns of variant polyadenylation signal usage in human genes. *Genome Research*, 10:1001–1010, 2000.

[25] J.S. Beckmann, V. Brendel, and E.N. Trifonov. Intervening sequences exhibit distinct vocabulary. *Journal of Biomolecular Structure and Dynamics*, 4(3):391–400, 1986.

[26] M. Bekaert. Recherche de décalages de phase de lecture chez *Saccharomyces cerevisiae*. Rapport de stage de DEA "Physiologie et Génétique des Microorganismes", université Paris-Sud XI, juin 2001.

[27] C.J. Benham. Computation of DNA structural variability – a new predictor of DNA regulatory regions. *Comput. Appl. Biosci.*, 12:375–381, 1996.

[28] G. Benson. Tandem repeats finder: a program to analyse DNA sequences. *Nucleic Acids Research*, 27(2):573–580, 1999.

[29] A. Bertoni, M. Goldwurm, and M. Santini. Random generation and approximate counting of ambiguously described combinatorial structures. In *Proceedings of STACS'00*, volume 1770 of *Lecture Notes in Computer Science*, pages 567–580, 2000.

[30] J. Bétréma and J.G. Penaud. Modèles avec particules dures, animaux dirigés et séries en variables partiellement commutatives. Technical Report 93-18, LaBRI, Université Bordeaux I, 1993.

[31] B. Bollobás. *Random graphs*. Academic Press, 1985.

[32] A. Bolshoy, P. McNamara, R.E. Harrington, and E.N. Trifonov. Curved DNA without A-A: experimental estimation of all 16 DNA wedge angles. *Proceedings of the National Academy of Science USA*, 88:2312–2316, 1991.

[33] A. Bolshoy and E. Nevo. Ecologic genomics of DNA: upstream bending in prokaryotic promoters. *Genome Research*, 10:1185–1193, 2000.

[34] M. Bousquet-Mélou and A. Conway. Enumeration of directed animals on an infinite family of lattices. *J. Phys. A: Math. Gen.*, 29:3357–3365, 1996.

[35] J.V. Braun and H.G. Müller. Statistical methods for DNA sequence segmentation. *Statistical Science*, 13(2):142–162, 1998.

[36] V. Brendel, J.S. Beckmann, and E.N. Trifonov. Linguistics of nucleotide sequences: morphology and comparison of vocabularies. *Journal of Biomolecular Structure and Dynamics*, 4(1):11–21, 1986.

[37] I. Brierley. Ribosomal frameshifting on viral RNAs. *Journal of General Virology*, 76(8):463–479, 1995.

[38] I. Brierley, A. Jenner, and S.C. Inglis. Mutational analysis of the "slippery sequence" component of a coronavirus ribosomal frameshifting signal. *Journal of Molecular Biology*, 227(2):463–479, 1992.

[39] A. Broder. Generating random spanning trees. In *30th Annual Symposium on Foundations of Computer Science*, pages 442–447. IEEE, 1989.

[40] Ph. Bucher. Regulatory elements and expression profiles. *Current Opinion in Structural Biology*, 9:400–407, 1999.

[41] Ph. Bucher and A. Bairoch. A generalised profile syntax for biomolecular sequence motifs and its function in automatic sequence interpretation. In *ISMB-94*, pages 53–61. AAAI Press, 1994.

[42] Ph. Bucher, K. Karplus, N. Moeri, and K. Hofmann. A flexible motif search technique based on generalised profiles. *Computers and Chemistry*, 20(1):3–23, 1996.

[43] M. Burset and R. Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34:353–367, 1996.

[44] J.-C. Callen. *Biologie cellulaire, des molécules aux organismes*. Dunod, 1999.

[45] G.A. Churchill. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51(1):79–94, 1989.

[46] C.J. Colbourn, W.J. Myrvold, and E. Neufeld. Two algorithms for unranking arborescences. *Journal of Algorithms*, 1995.

[47] R. Cori and B. Vauquelin. Planar maps are well labeled trees. *Canadian Journal of Mathematics*, 33(5):1023–1042, 1981.

[48] S. Corteel, A. Denise, I. Dutour, F. Sarron, and P. Zimmermann. CS web page. http://dept-info.labri.u-bordeaux.fr/∼dutour/CS/.

[49] S. Corteel, A. Denise, and D. Gouyou-Beauchamps. Bijections for directed animals on infinite families oflattices. *Annals of Combinatorics*, 4:271–286, 2000.

[50] S. Corteel, D. Gardy, M. Valencia-Pabon, D. Barth, and A. Denise. The permutation-path coloring problem on trees. *Theoretical Computer Science*, 2001. 25 pages. A paraître.

[51] E. Coward. Shufflet: Shuffling sequences while conserving the k-let counts. *Bioinformatics*, 15:1058–1059, 1999. Programme : http://www.genetique.uvsq.fr/eivind/shufflet.html.

[52] J. Craig Venter et al. The sequence of the human genome. *Science*, 291:1304–1351, 2001.

[53] M. Crochemore, C. Hancart, and T. Lecroq. *Algorithmique du texte*. Vuibert, 2001.

[54] M. Crochemore and W. Rytter. *Text algorithms*. Oxford University Press, 1994.

[55] D. Daney, G. Hanrot, V. Lefèvre, F. Rouillier, and P. Zimmermann. The MPFR library. http://www.mpfr.org.

[56] M. Daumas and J.-M. Muller. *Qualité des calculs sur ordinateur. Vers des arithmétiques plus fiables ?* Masson, 1997.

[57] M. Delest and X.G. Viennot. Algebraic languages and polyominoes enumeration. *Theoretical Computer Science*, 34:169–206, 1984.

[58] A. Denise. Generation of random planar maps. In G. Di Battista, P. Eades, H. de Fraysseix, P. Rosenstiehl, and R. Tamassia, editors, *Proceedings of Graph Drawing'93*, page 87, 1993.

[59] A. Denise. *Méthodes de génération aléatoire d'objets combinatoires de grande taille et problèmes d'énumération*. PhD thesis, Université Bordeaux I, 1994.

[60] A. Denise. Génération aléatoire et uniforme de mots. *Discrete Mathematics*, 153:69–84, 1996.

[61] A. Denise. Génération aléatoire et uniforme de mots de langages rationnels. *Theoretical Computer Science*, 159(1):43–63, 1996.

[62] A. Denise, M. Bekaert, L. Bidou, G. Duchateau-Nguyen, J.-P. Forest, Ch. Froidevaux, I. Hatin, J.-P. Rousset, and M. Termier. Vers une modélisation des sites de décalage de cadre en -1 dans les génomes eucaryotes. In L. Duret, C. Gaspin, and T. Schiex, editors, *Actes de JOBIM'2001*, pages 229–237, 2001.

[63] A. Denise, Ch. Dürr, and F. Ibn-Majdoub-Hassani. Enumération et génération aléatoire de polyominos convexes en réseau hexagonal. In *Actes de FPSAC'97*, pages 222–234, 1997.

[64] A. Denise, I. Dutour, and P. Zimmermann. CS: a MuPAD package for counting and randomly generating combinatorial structures. In *Actes de FPSAC'98*, pages 195–204, 1998. Publié aussi dans MathPAD 8 (1) 1998 22–29, http://www.mupad.de/mathpad.shtml.

[65] A. Denise, M. Régnier, and M. Vandenbogaert. Assessing statistical significance on overrepresented oligonucleotides. In O. Gascuel and B.E. Moret, editors, *Actes de WABI'01*, volume 2149 of *Lecture Notes in Computer Science*, pages 85–97. Springer, 2001.

[66] A. Denise, O. Roques, and M. Termier. Random generation of words of context-free languages according to the frequencies of letters. In D. Gardy and A. Mokkadem, editors, *Mathematics and Computer Science: Algorithms, Trees, Combinatorics and probabilities*, Trends in Mathematics, pages 113–125. Birkhaüser, 2000.

[67] A. Denise and R. Simion. Two combinatorial statistics on Dyck paths. *Discrete Mathematics*, 137:155–176, 1995.

[68] A. Denise, M. Vasconcellos, and D.J.A. Welsh. The random planar graph. *Congressus Numerantium*, 113:61–79, 1996.

[69] A. Denise and P. Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science*, 218:233–248, 1999.

[70] L. Devroye. *Non-uniform random variate generation*. Springer-Verlag, 1986.

[71] J.D. Dixon and H.S. Wilf. The random selection of unlabeled graphs. *Journal of Algorithms*, 4:205–213, 1983.

[72] S. Dong and D.B. Searls. Gene structure prediction by linguistic methods. *Genomics*, 23:540–551, 1994.

[73] M. Drmota. Systems of functional equations. *Random Structures and Algorithms*, 10:103–124, 1997.

[74] Ph. Duchon, Ph. Flajolet, G. Louchard, and G. Schaeffer. Random sampling from Boltzmann principles. Prépublication, décembre 2001.

[75] S. Dulucq and L. Tichit. À propos de la comparaison de structures secondaires. In L. Duret, C. Gaspin, and T. Schiex, editors, *Actes de JOBIM'2001*, pages 125–133, 2001.

[76] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis. Probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[77] I. Dutour and J.-M. Fédou. Object grammars and random generation. *Discrete Mathematics and Theoretical Computer Science*, 2:47–61, 1998.

[78] S.R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22:2079–2088, 1994.

[79] R.A. Elton. Theoretical models for heterogeneity of base composition in DNA. *Journal of Theoretical Biology*, 45:533–553, 1974.

[80] J.W. Fickett. Inferring genes from open reading frames. *Computers Chem.*, 18(3):203–205, 1994.

[81] J.W. Fickett and A.H. Hatzigeorgiou. Eukaryotic promoter recognition. *Genome Research*, 7:861–878, 1997.

[82] J.W. Fickett, D.C. Torney, and D.R. Wolf. Base compositional structure of genomes. *Genomics*, 13:1056–1064, 1992.

[83] W.M. Fitch. Random sequences. *Journal of Molecular Biology*, 163:171–176, 1983.

[84] Ph. Flajolet, Y. Guivarc'h, W. Spankowski, and B. Vallée. Hidden pattern statistics. In *Actes de ICALP'01*, volume 2076 of *Lecture Notes in Computer Science*, pages 152–165, 2001.

[85] Ph. Flajolet and R. Sedgewick. Analytic combinatorics: functional equations – rational and algebraic functions. Technical Report RR4103, INRIA, 2001.

[86] Ph. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132:1–35, 1994.

[87] Ph. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus of random generation: Unlabelled structures. Document non publié, 1997.

[88] H. Fleischner. *Eulerian graphs and related topics, Part 1*, volume 2 of *Annals of Discrete Mathematics*. North-Holland, 1991.

[89] J.-P. Forest. Reconnaissance de gènes à décalage de cadre dans les séquences d'ARN. Rapport de stage de DEA "Information, Interaction, Intelligence (I3)", université Paris-Sud XI, juin 2000.

[90] R. Gherbi and J. Hérisson. ADN Viewer: a software framework for 3D modeling and stereoscopic visualisation of the genome. In *Graphicon'2000*, Moscou, 2000.

[91] R. Gherbi and J. Hérisson. 3D modeling tools for spatial-based in silico analysis of DNA. *International Electronic Journal on Computer Graphics and Geometry*, 3(1), 2001.

[92] M. Goldwurm. Random generation of words in an algebraic language in linear binary space. *Information Processing Letters*, 54:229–233, 1995.

[93] V. Gore, M. Jerrum, S. Kannan, Z. Sweedyk, and S. Mahaney. A quasi-polynomial-time algorithm for sampling words from a context-free language. *Information and Computation*, 134(1):59–74, 1997.

[94] S.-D. Gouraud. Application des techniques de génération aléatoire au test de logiciel, janvier 2001. Rapport de stage du D.E.A. Programmation, version révisée.

[95] S.-D. Gouraud, A. Denise, M.-C. Gaudel, and B. Marre. A new way for automating statistical testing methods. In *Actes de ASE'01 (IEEE International Conference on Automated Software Engineering*, San Diego, novembre 2001. A paraître.

[96] D. Gouyou-Beauchamps. Standard Young tableaux of heigth 4 and 5. *European Journal of Combinatorics*, 10:69–82, 1989.

[97] D.H. Greene and D.E. Knuth. *Mathematics for the analysis of algorithms.* Birkhäuser, 1981.

[98] M. Gribskov, R. Lüthy, and D. Eisenberg. Profile analysis. *Methods in Enzymology*, 183:146–159, 1990.

[99] M. Gribskov, A.D. MacLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Science USA*, 84:4355–4358, 1987.

[100] L. Gueguen. Segmentation by maximal predictive partitioning according to composition biases. In O. Gascuel and M.-F. Sagot, editors, *Computational Biology. Sélection d'articles de JO-BIM'00*, volume 2066 of *Lecture Notes in Computer Science*, pages 32–44. Springer, 2001.

[101] A. Guénoche. Random spanning tree. *Journal of Algorithms*, 4:214–220, 1983.

[102] R. Guigó, S. Knudsen, N. Drake, and T.F. Smith. Prediction of gene structure. *Journal of Molecular Biology*, 226:141–157, 1992.

[103] D. Gusfield. *Algorithms on strings, trees and sequences. Computer science and computational biology.* Cambridge University Press, 1997.

[104] A.B. Hammel, R.C. Taylor, S.W. Peltz, and J.D. Dinman. Identification of putative programmed -1 ribosomal frameshift signals in large DNA databases. *Genome Research*, 9(5):417–427, 1999.

[105] T. Hickey and J. Cohen. Uniform random generation of strings in a context-free language. *SIAM J. Comput.*, 12(4):645–655, 1983.

[106] W. Hochstättler, M. Loebl, and Ch. Moll. Generating convex polyominoes at random. *Discrete Mathematics*, 153:165–176, 1996.

[107] K. Hofmann and Ph. Bucher. Methods for the construction of profile entries for the PROSITE database. Poster présenté à ISMB'95. Document accessible à l'adresse http://www.isrec.isb-sib.ch/profile/ismb95/poster.html.

[108] K. Hofmann, Ph. Bucher, L. Falquet, and A. Bairoch. The PROSITE database, its status in 1999. *Nucleic Acids Research*, 27(1):215–219, 1999.

[109] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[110] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989.

[111] M. Jerrum and A. Sinclair. The Markov chain Monte-Carlo method: an approach to approximate counting and integration. In D.S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 482–520. PWS Publishing, 1996.

[112] M. Jerrum, L.G. Valiant, and V.V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[113] D. Kandel, Y. Matias, R. Unger, and P. Winkler. Shuffling biological sequences. *Discrete Applied Mathematics*, 71:171–185, 1996.

[114] H. Karas, R. Knuppel, W. Schulz, H. Sklenar, and E. Wingender. Combining structural analysis of DNA with search routines for the detection of transcription regulatory elements. *Comput. Appl. Biosci.*, 12:375–381, 1996.

[115] M. Klaerr-Blanchard, H. Chiapello, and E. Coward. Detecting localized repeats in genomic sequences: A new strategy and its application to *B. subtilis* and *A. thaliana*. *Comput. Chem.*, 24(1):57–70, 2000.

[116] D.E. Knuth. Mathematical analysis of algorithms. *Information Processing*, 71:19–27, 1972.

[117] A. Krogh. An introduction to hidden Markov models for biological sequences. In S.L. Salzberg, D.B. Searls, and S. Kasif, editors, *Computational methods in molecular biology*, volume 32 of *New comprehensive biochemistry*, pages 45–63. Elsevier, 1998.

[118] F. Lefebvre. An optimized parsing algorithm well suited to RNA folding. In C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lenguauer, and S. Wodak, editors, *ISMB'95*, pages 222–230. AAAI press, 1995.

[119] F. Lefebvre. A grammar-based unification of several alignment and folding algorithms. In D.J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R.F. Smith, editors, *ISMB'96*, pages 143–154. AAAI press, 1996.

[120] J. Liphardt. *The mechanism of* -1 *ribosomal frameshifting: experimental and theoretical analysis.* PhD thesis, Churchill College, 1999.

[121] D.J. Lipman and W.R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435–1441, 1985.

[122] D.J. Lipman, W.J. Wilbur, T.F. Smith, and M.S. Waterman. On the statistical signifiance of nucleic acid similarities. *Nucleic Acids Research*, 12:215–226, 1984.

[123] T.M. Lowe and S.R. Eddy. tRNAscan-SE: a program for improved detection of transfer tRNA genes in genomic sequence. *Nucleic Acids Research*, 25:955–969, 1997.

[124] A.V. Lukashin and M. Borodovski. GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Research*, 26(4):1107–1115, 1998.

[125] H.G. Mairson. Generating words in a context free language uniformly at random. *Information Processing Letters*, 49:95–99, 1994.

[126] V. Makeev, V. Ramensky, M. Gelfand, M. Roytberg, and V. Tumanyan. Bayesian approach to DNA segmentation into regions with different average nucleotide composition. In O. Gascuel and M.-F. Sagot, editors, *Computational Biology. Sélection d'articles de JOBIM'00*, volume 2066 of *Lecture Notes in Computer Science*, pages 57–73. Springer, 2001.

[127] B. Marczinke, R. Fisher, M. Vidakovic, A. Bloys, and L. Brierley. Secondary structure and mutational analysis of the ribosomal frameshift signal of Rous sarcoma virus. *Journal of Molecular Biology*, 284(2):205–225, 1998.

[128] M. Marilley and P. Pasero. Common DNA structural features exhibited by eukaryotic ribosomal gene promoters. *Nucleic Acids Research*, 24(12):2204–2211, 1996.

[129] B. Marre and A. Arnould. Test sequences generation from lustre descriptions : Gatel. In *15h I.E.E.E. International Conference on Automated Software Engineering*, pages 229–237, 2000.

[130] F. Muri-Majoube and B. Prum. Une approche statistique de l'analyse des génomes. *La Gazette des Mathématiciens*, 89:63–98, 2001.

[131] P. Nicodème, B. Salvy, and Ph. Flajolet. Motif statistics. *Theoretical Computer Science*, 2001. A paraître.

[132] A. Nijenhuis and H.S. Wilf. *Combinatorial algorithms.* Academic Press Inc., 1979.

[133] G. Nuel. *Grandes déviations et chaînes de Markov pour l'étude des occurrences de mots dans les séquences biologiques.* PhD thesis, Université d'Evry Val d'Essonne, 2001.

[134] S.G. Oliver et al. The complete DNA sequence of yeast chromosome III. *Nature*, 357:38–46, 1992.

[135] E. Passarge. *Atlas de poche de génétique.* Flammarion, 1995.

[136] N. Pavy, S. Rombauts, P. Déhais, C. Mathé, D.V.V. Ramana, Ph. Leroy, and P. Rouzé. Evaluation of gene prediction software using a genomic data set: application to *Arabidopsis thaliana* sequence. *Bioinformatics*, 15(11):887–899, 1999.

[137] A.G. Pedersen, P. Baldi, Y. Chauvin, and S. Brunak. DNA structure in human RNA polymerase II promoters. *Journal of Molecular Biology*, 281:663–673, 1998.

[138] J.-G. Penaud and O. Roques. Tirage à pile ou face de mots de Fibonacci. In *Actes du colloque LaCIM 2000*, Publications du LACIM, 2000.

[139] Y. Ponty. Génération aléatoire de structures génomiques : GenRGenS. Rapport de TER-stage, maîtrise d'informatique, université Paris-Sud XI, juin 2001.

[140] J.G. Propp. Generating random elements of a finite distributive lattice. *Electronic Journal of Combinatorics*, 4(2), 1997. R15.

[141] J.G. Propp and D.B. Wilson. How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms*, 27(2):170–217, May 1998.

[142] B. Prum, F. Rodolphe, and É. de Turckheim. Finding words with unexpected frequencies in DNA sequences. *J. Royal Statist. Soc. Ser. B*, 57:205–220, 1995.

[143] M.G. Reese, F.H. Eeckman, D. Kulp, and D. Haussler. Improved splice site detection in Genie. *Journal of Computational Biology*, 4:311–323, 1997.

[144] M.G. Reese, G. Hartzell, N.L. Harris, U. Ohler, J.F. Abril, and S.E. Lewis. Genome annotation assesment in *Drosophila melanogaster*. *Genome research*, 10:483–501, 2000.

[145] M. Régnier. A unified approach to word occurrence probabilities. *Discrete Applied Mathematics*, 104(1):259–280, 2000.

[146] M. Régnier and A. Denise. Word statistics conditioned by overrepresented words. En préparation. Version actuelle disponible à http://algo.inria.fr/regnier, 2001.

[147] M. Régnier and W. Szpankowski. On pattern frequency occurrences in a Markovian sequence. *Algorithmica*, 22(4):631–649, 1998.

[148] G. Reinert, S. Schbath, and M.S. Waterman. Probabilistic and statistical properties of words: An overview. *Journal of Computational Biology*, 7(1–2):1–46, 2000.

[149] J.-L. Rémy. Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *R.AI.R.O Informatique Théorique*, 19(2):179–195, 1985.

[150] E. Rivas and R. Eddy. The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics*, 16, 2000.

[151] R. Rivière. Graphes eulériens et séquences génomiques aléatoires sous contraintes. Rapport de stage de DEA "Mathématiques et Informatique", universités Versailles-St-Quentin et Paris-Sud XI, septembre 2001.

[152] J. de Rumeur. *Communications dans les réseaux de processus*. Masson, 1994.

[153] M.-F. Sagot. L'espoir est cette chose avec des plumes – Algorithmes combinatoires et biologie moléculaire. Habilitation à diriger les recherches, 2000.

[154] M.-F. Sagot. Communication lors de la journée de l'action REMAG, Institut Pasteur, janvier 2001.

[155] Y. Sakakibara, M. Brown, R. Hughey, I.S. Mian, K. Sjölander, R.C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22:5112–5120, 1994.

[156] F. Sarron. GenRGenS v0.6 : notes de développement et de maintenance. Rapport de stage, université Paris-Sud XI, septembre 2000.

[157] G. Schaeffer. *Conjugaison d'arbres et cartes combinatoires aléatoires*. PhD thesis, Université Bordeaux I, 1998.

[158] G. Schaeffer. Random sampling of large planar maps and convex polyhedra. In *Proceedings of STOC'99*, Atlanta, 1999.

[159] S. Schbath. *Etude asymptotique du nombre d'occurrences d'un mot dans une chaîne de Markov et application à la recherche de mots de fréquence exceptionnelle dans les séquences d'ADN.* PhD thesis, Université paris V, 1995.

[160] S. Schbath. An efficient statistic to detect over- and under-represented words in DNA sequences. *Journal of Computational Biology*, pages 189–192, 1997.

[161] D.B. Searls. The linguistics of DNA. *American Scientist*, 80(6):579–591, 1992.

[162] D.B. Searls. String variable grammar: a logic grammar formalism for the biological language of DNA. *Journal of Logic Programming*, 24(1,2):73–102, 1995.

[163] D.B. Searls. Linguistic approaches to biological sequences. *CABIOS*, 13(4):333–344, 1997.

[164] D.B. Searls. Formal language theory and biological macromolecules. *Series in Discrete Mathematics and Theoretical Computer Science*, 47:117–140, 1999.

[165] A. Sinclair. *Algorithms for Random Generation and Counting, a Markov Chain Approach.* Birkhaüser, 1992.

[166] R. Staden. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Research*, 12:505–519, 1984.

[167] G. Stahl, L. Bidou, J.-P. Rousset, and M. Cassan. Versatile vectors to study recoding: conservation of rules between yeast and mammalian cells. *Nucleic Acids Research*, 23:1557–1560, 1995.

[168] G. Stahl and J.-P. Rousset. Les surprises du décodage de l'information génétique. *Médecine/Sciences*, 15:1118–1125, 1999.

[169] G.D. Stormo. Gene-finding approaches for eukaryotes. *Genome Research*, 10:394–397, 2000.

[170] M. Termier and A. Kalogeropoulos. Discrimination between fortuitous and biologically constrained open reading frames in DNA sequences of *S. cerevisiae*. *Yeast*, 12:369–384, 1996.

[171] P. Thévenod-Fosse and H. Waeselynck. An investigation of software statistical testing. *The Journal of Software testing, Verification and reliability*, 1(2):5–26, 1991.

[172] G. Tinhofer. On the generation of random graphs with given properties and known distribution. *Appl. Comput. Sci. Ber. Prakt. Inf.*, 13:265–296, 1979.

[173] G. Tinhofer. On the use of almost sure graph properties. *Lecture Notes in Computer Science*, 100:113–126, 1980.

[174] G. Tinhofer. Generating graphs uniformly at random. *Computing Supplementum*, 7:235–255, 1990.

[175] E.N. Trifonov. The multiple codes of nucleotide sequences. *Bull. Math. Biol.*, 51(4):417–432, 1989.

[176] E.N. Trifonov. Communication personnelle, 2000.

[177] E.N Trifonov and J.L. Sussman. The pitch of chromatin DNA is reflected in its nucleotide sequence. *Proceedings of the National Academy of Science USA*, 77:3816–3820, 1980.

[178] W. T. Tutte. A census of planar maps. *Canadian Journal of Mathematics*, 15:249–271, 1963.

[179] B. Van Cutsem. Les structures combinatoires en classification. In *Actes de GASCom 94*, Bordeaux, 1994.

[180] J. van der Hoeven. Relax, but don't be too lazy. Technical Report 78, Université de Paris-Sud, Mathématiques, Bâtiment 425, F-91405 Orsay, 1999. Document accessible à `http://www.math.u-psud.fr/~vdhoeven`.

[181] A. Vanet, L. Marsan, and M.-F. Sagot. Promoter sequences and algorithmical methods for identifying them. *Res. Microbiol.*, 150:779–799, 1999.

[182] M. Vauchaussade de Chaumont and X.G. Viennot. Enumeration of RNA's secondary structures by complexity. In V. Capasso, E. Grosso, and S.L. Paven-Fontana, editors, *Mathematics in Medecine and Biology*, volume 57 of *Lecture Notes in Biomathematics*, pages 360–365, 1985.

[183] X. G. Viennot. Problèmes combinatoires posés par la physique statistique. *Astérisque*, 121-122:225–246, 1985. Séminaire Bourbaki 626, $26^{ème}$ année.

[184] H.S. Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, 24:281–291, 1977.

[185] H.S. Wilf. Récents développements et problèmes dans le domaine de la génération aléatoire. *Theoretical Computer Science*, 159(1):5–13, 1996.

[186] D.B. Wilson. Web site for perfectly random sampling with Markov chains. Adresse : http://dimacs.rutgers.edu/∼dbwilson/exact.

[187] D.B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 296–303, 1996.

[188] D.S. Wilson and J.W. Szostak. In vitro selection of functional nucleic acids. *Annu. Rev. Biochem.*, 68:611–647, 1999.

[189] N. C. Wormald. Generating random unlabelled graphs. *SIAM J. Comput*, 16(4):717–727, 1987.

[190] B. Ycart. Stopping tests for Markov chain Monte-Carlo methods. *Methods and Computing in Applied Probabilities*, 2(1):23–36, 2000.

[191] E. Yeramian. Genes and the physics of the DNA double-helix. *Gene*, 255(2):139–150, 2000.

[192] E. Yeramian. The physics of DNA and the annotation of the *Plasmodium falciparum* genome. *Gene*, 255(2):151–168, 2000.

[193] Y.-T. Yu, R.E. Breitbart, L.B. Smoot, Y. Lee, V. Mahdavi, and B. Nadal-Ginard. Human myocite-specific enhancer factor 2 comprises a group of tissue-restricted mads box transcription factors. *Genes and Development*, 6:1783–1798, 1992.

[194] P. Zimmermann. Gaïa: a package for the random generation of combinatorial structures. *MapleTech*, 1(1):38–46, 1994.

Index