THÈSE DE DOCTORAT
en Informatique

Inférence de motifs structurés : algorithmes et outils appliqués à la détection de sites de fixation dans des séquences génomiques

Laurent Marsan

Numéro officiel
Université de Marne-la-Vallée

Thèse de doctorat

Spécialité : Informatique

présentée par

Laurent Marsan

pour l'obtention du titre de

Docteur en Informatique

sur le sujet

Inférence de motifs structurés : algorithmes et outils appliqués à la détection de sites de fixation dans des séquences génomiques

directeurs de thèse

Pr. Maxime Crochemore
Pr. Marie-France Sagot

soutenue le 8 Avril 2002 devant le jury composé de :

Pr. Jean Berstel (Examinateur)
Pr. Serge Dulucq (Rapporteur)
Dr. Roberto Grossi (Rapporteur)
Dr. Anne Vanet (Invitée)
Dr. Alain Viari (Rapporteur)

à mon oncle, Ely
et à Mister Jey qui manquent à bord

Merci !

Cette thèse a débuté il y a près de quatre ans, durant lesquels j'ai navigué entre mes « terres d'origine » (l'Université de Marne-la-Vallée) et l'Institut Pasteur. Au cours d'une si longue période, on rencontre nécessairement (et on perd parfois de vue) nombre de personnes. Je me rends compte aujourd'hui en écrivant ces lignes que cette thèse a été aussi enrichissante humainement que scientifiquement.

Marie-France est bien sûr la personne que j'ai le plus côtoyée durant ce travail, dont je ne serais assurément jamais venu à bout si elle n'avait fait preuve d'autant de patience et de compréhension. Merci Marie-France, de ton aide constante, de ton soutien et d'avoir partagé un peu de ta passion.

Je remercie Maxime Crochemore d'avoir accepté d'encadrer cette thèse, de sa disponibilité, de sa curiosité, de sa spontanéité, et de sa gentillesse.

Merci à Serge Dulucq et à Alain Viari d'avoir accepté de rapporter cette thèse, d'avoir consacré du temps à sa lecture assidue, et de leurs appréciations.

Grazie molte a Roberto Grossi per la sua gentilezza, per la sua spontaneità, e per avere accettato di giudicare il mio lavoro. Pronto per il Marocco ?

Merci à Jean Berstel d'avoir spontanément manifesté de l'intérêt pour ces travaux et accepté de participer au jury.

Merci à Anne Vanet pour sa spontanéité, sa franchise, son humour, sa collaboration, ses conseils et sa participation au jury.

Merci à Karl Forner et à Étienne Duris, pour avoir fait l'effort de lire et de comprendre ce travail et m'avoir fourni des remarques sincères et pertinentes. Merci également à Anne Cariou, Corinne Maufrais, Raquel Tavares et Sophie Creno pour leurs corrections.

Cette thèse a, comme toutes les thèses parait-il, fait l'objet de pas mal de crises de nerf, chutes de moral et remises en question. Un gros merci à ceux de l'IGM qui m'ont aidé quand j'en avais besoin, par leur écoute, leur attention, leur compétence ou une bière fraîche, chacun à sa façon : Andrée, Anne, Dominique, Elsa, Étienne, Fred, Gilles, Giuseppina, Line, Marc, Nicolas et Olivier. Merci également à tous ceux du labo qui contribuent à ce que l'ambiance y soit ce qu'elle est aujourd'hui : chaleureuse. Ce fût une joie de travailler parmi vous.

Côté Pasteur, je remercie chaleureusement Daniel Azuelos, Laurent Bloch, Bernard Caudron et Gérard Masson de m'avoir accueilli et laissé m'étaler sur un des immenses bureaux du Service d'Informatique Scientifique de l'Institut. Mes remerciements vont également aux membres du SIS qui m'ont accueilli et aidé dans mon travail : Babafou, Catherine, Corinne, Élodie, Éric, Frédérique, Katja, Louis, Marie-Christine, Michel, Olivier, Nicolas, Stéphane, Tru. Merci également à Muriel Robert et à Mme Nau pour leur aide et leur gentillesse.

Merci et bonne route aux thésards, futurs thésards et ex-thésards rencontrés en chemin qui m'ont aidé ou ont collaboré à ce travail : Anne Cariou, Chloé Rispal, Ingrid Lafontaine, Julien Allali, Marina Zelwer, Mathieu Raffinot, Philippe Blayo, Seb du Québec (pour une fructueuse collaboration sur l'étude du dentifrice bicolore), Sophie Creno et Vince Lemaout. "Spécial dédicace" to Rémi Forax pour...pas mal de choses !

Grazie mille a Nadia Pisanti per il suo aiuto, la sua collaborazione, i suoi consigli, e per avermi accolto in modo caloroso.

Grazie a Fabrizio Luccio e ai membri del laboratorio di Informatica de l'Università di Pisa che mi hanno accolto per più de due mesi.

Merci à Mathias Vandenbogaert et à Mireille Régnier pour le temps qu'ils m'ont consacré, l'intérêt porté à mes travaux, leur aide et leur collaboration.

Thanks to Gert Thijs, for the experiments he made for me with Motifs Sampler, and for sharing his point of view about motifs inference problem and solutions.

Après tant d'années d'études dans ce domaine, je suis maintenant définitivement convaincu que l'informatique est une science occulte. Aussi, je me dois de remercier les quelques magiciens, gourous et autres shamans qui ont exorcisé ma bécane ou mes programmes tout au long de cette thèse, en particulier : Daniel Azuelos, Patrice Hérault (adepte de la méthode dure, dite "wouaaa...poum !"), encore Rémi Forax (les *templates* en C, c'est possible), Éric Deveaud, Babafou, Nicolas Joly, Olivier Perret, Stéphane Bortzmeyer, Julien Allali, Jérôme-qui-tape-vite et Bud.

Merci aux béta-testeurs, ceux qui se sont cassés les dents sur les version béta-boîteuses de SMILE : Élodie Guillaume et Yves Daubenton, Éric Rivals, et particulièrement Anne Cariou qui, forte de ses deux mois de vol sur SMILE, obtient de loin la palme du plus grand nombre de vilains bugs débusqués.

Merci à Marie-Odile Delorme, dont le passionnant cours de bio en DEA a aiguisé mon envie de travailler dans ce domaine (et s'est avéré être d'une aide précieuse lors de la rédaction de l'intro biologique de cette thèse).

Des remerciements et mon amitié à tous les « copains d'abord », de Marne à Ménilmontant, pour beaucoup rencontrés à la fac, qui m'ont soutenu et encouragé (« alors Lolo, t'as trouvé ? ») durant ces années. Tout le monde a quitté l'université dont je sors bon dernier, mais la petite bande qui s'y est construite pendant les joyeuses années du D2 est maintenant une famille.

Merci à Annette, plein de bonheur à toi.

Merci à Karl pour son soutien moral, sportif, scientifique et technique.

Merci à Julien, amigo eh !

Enfin, merci à mon frère, à mes parents et à mes grands-parents pour leurs attentions et leur soutien sans faille.

Résumé

Au cours de ce travail, nous nous sommes particulièrement intéressés à un problème de biologie moléculaire : la détection de sites de fixation dans des séquences d'ADN. Ce problème, perçu sous un certain angle, trouve des solutions variées grâce aux travaux réalisés en algorithmique du texte. Après une présentation des spécificités de ces sites, nous passons en revue les représentations informatiques existantes utilisées pour les modéliser. Puis, nous faisons état des différents travaux algorithmiques effectués dans le domaine de leur détection. La pertinence des principales approches est discutée. Nous essayons en particulier de présenter les différents aspects du compromis qui paraît inévitable entre sensibilité et complexité quand on traite un tel problème.

Notre apport consiste ensuite à développer une nouvelle représentation pour les sites de fixation. Celle-ci prend en compte une caractéristique de certains d'entre eux : leur capacité à s'associer sous certaines contraintes. Nous introduisons la notion de modèle structuré, et développons plusieurs algorithmes combinatoires exacts de détection de tels modèles. Nous présentons ensuite l'outil que nous avons conçu à partir de ces algorithmes, dénommé SMILE. Nous ramenant au problème biologique qui a motivé ces travaux algorithmiques, nous appliquons cet outil à l'inférence de sites de fixation connus et inconnus dans des jeux de séquences nucléiques expérimentaux ou directement issus de génomes complets. Les résultats de ces expériences sont comparés avec ceux qu'obtiennent certains outils couramment utilisés sur les mêmes jeux de données, et leur pertinence biologique est discutée. Pour finir, nous jugeons l'apport des modèles structurés et esquissons plusieurs directions à explorer pour améliorer la détection de sites de fixation.

Les représentations, algorithmes et outils développés dans cette thèse sont généraux, et peuvent donc être appliqués à l'extraction de tous types de signaux structurés et approchés, communs à plusieurs séquences. En particulier, ils peuvent d'ores et déjà être utilisés pour inférer des motifs dans des séquences protéiques.

Abstract

In this work, we concentrated our interest on a problem in molecular biology : the detection of binding sites in DNA sequences. This problem, from a certain point of view, finds diverse solutions in text algorithmics. We first present the specificity of such sites, and look at the existing representations used to modelize them. We then review the existing algorithms aimed at detecting these sites, and try to evaluate the pertinence of the main approaches employed. In particular, we try to discuss the trade off that exists between sensibility and complexity when dealing with such a problem.

Our work consists in developing a new representation for binding sites, which incorporates one of the main characteristics of some of them : their ability to appear associated in a more or less constrained manner. We introduce the notion of a structured model, develop several exact combinatorial algorithms to infer such models, and present the software, called SMILE, we made using these algorithms. Going back to the biological problem which motivated this work, we apply our tool to infer known and unknown binding sites in a few sets of DNA sequences. The results of these experiments are compared to those obtained by some of the most used software on the same sets of sequences. We then discuss of the biological pertinence of all these results. Finally, we try to place our work in the current context, and define several directions to explore to improve the inference of binding sites.

The algorithms and tools we made are all generic, they can be applied to extract any kind of structured signals that are common to a set of sequences. In particular, they can already treat protein sequences.

Table des matières

Table des figures

Liste des tableaux

Introduction

De ses balbutiements avec les expériences de Gregor Mendel en 1865, au séquençage complet du génome humain qui devrait être totalement achevé d'ici un à deux ans, la génétique a véritablement bouleversé la biologie en l'espace d'un siècle et demi, apportant la rigueur des mathématiques à une science jusque là descriptive. Répondant à des questions sur lesquelles l'espèce humaine s'interroge depuis des millénaires, les découvertes faites en génétique ont eu des répercussions considérables dans nombre de domaines, offrant ainsi une nouvelle perception du monde et de la place que nous y occupons.

La biologie moléculaire naît de la génétique au milieu du XX$^{\text{ème}}$ siècle. Les premières expériences visant à mettre en évidence des mécanismes moléculaires - à l'échelle des atomes - impliqués dans le fonctionnement d'organismes vivants sont réalisées par l'équipe d'Alfred Hershey, en 1952. Dès lors, la biologie moléculaire connaît un essor fantastique, dont le véritable signal de départ est donné par la découverte de la structure de la molécule d'*acide désoxyribo-nucléique* (ADN), support de l'information génétique de tous les êtres vivants, par James Watson et Francis Crick en 1953. Cette découverte initie la modélisation des mécanismes génétiques à l'échelle moléculaire qui se poursuit à ce jour.

Le séquençage des macro-molécules d'ADN est un procédé consistant à déchiffrer l'enchaînement de nucléotides (les constituants de l'ADN) qui composent les génomes d'organismes types, dont celui de l'homme. Il est le fruit des nombreuses avancées scientifiques et techniques qui ont suivi la naissance de la biologie moléculaire. Des banques de données informatiques stockent et répertorient les séquences nucléiques obtenues par différents organismes de recherche à travers le monde. Depuis vingt ans, la taille de la plus grande d'entre elles, celle de l'EMBL (*European Molecular Biology Laboratory*), et des autres banques du même type, suit une courbe de croissance exponentielle (voir la figure 1). Cette explosion de la quantité de données génétiques disponibles n'a cependant pas été accompagnée d'une augmentation comparable de la capacité d'analyse de ces données. À ce jour, si leur acquisition n'est plus une difficulté majeure, le principal problème est donc de comprendre ces données génétiques, qui sont maintenant rendues disponibles en masse par la généralisation et l'accélération du séquençage des génomes.

C'est ce constat qui fait se rencontrer, il y a une vingtaine d'années, biologie moléculaire et informatique, donnant naissance à un domaine nouveau : la bio-informatique. Face à la quantité de données à traiter, et ayant besoin de systématiser les procédés d'analyse de façon efficace, les biologistes moléculaires n'ont d'autre choix que d'avoir recours à l'infor-

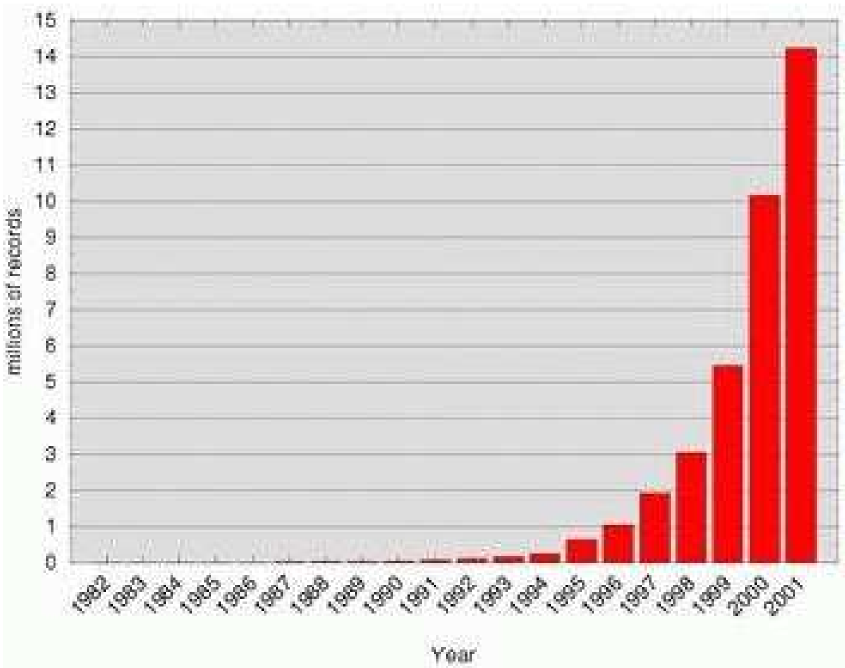

Fig. 1 – Nombre de séquences (en millions) enregistrées dans la base EMBL depuis 1982. (Source : EMBL, Décembre 2001)

matique. De l'autre côté, des mathématiciens et informaticiens attirés par les nombreux problèmes attrayants ouverts par cette science naissante, et forts de leurs expériences dans des domaines tels que l'algorithmique, les statistiques, ou les probabilités, orientent leurs recherches vers la biologie moléculaire.

Considérant une séquence d'ADN comme un texte sur un alphabet à quatre symboles (A, C, G et T), l'algorithmique du texte trouve dans les problèmes posés en biologie moléculaire de vastes sujets de recherche, dont certains sont très proches de problèmes algorithmiques plus anciens que la bio-informatique. Les algorithmiciens se tournant vers la bio-informatique se trouvent confrontés à l'étude d'un nouveau langage, dont l'alphabet est certes défini mais dont on ne connaît pas la syntaxe ni la grammaire. Ils disposent cependant d'un atout de taille par rapport aux archéologues découvrant une langue ancienne : le langage étudié a toujours cours aujourd'hui, et de nombreuses expériences sont possibles sur ces macro-molécules, qui permettent de fournir des informations sur leurs fonctions biologiques, et de révéler ainsi, petit à petit, l'organisation de l'information contenue dans les séquences biologiques.

Dans ce contexte, nos travaux se sont construits autour d'un problème biologique précis, concernant des zones particulières d'un génome qui contiennent des éléments du langage génétique ayant un rôle caractéristique : les *sites de fixation des facteurs de transcription*. L'objectif biologique de nos travaux était de fournir aux bio-informaticiens des outils permettant l'*inférence*, c'est-à-dire la détection, efficace de ces sites.

Perçu sous un certain angle, ce problème biologique s'apparente à un problème bien connu de l'algorithmique du texte, l'inférence de motifs communs à plusieurs textes, qui nécessite cependant d'être adapté à la complexité et à la variabilité des sites de fixation. La plupart des algorithmes essayant de résoudre ce problème de manière exacte utilisent des motifs sans erreurs ([Waterman, 1984], [Staden, 1989], [van Helden et al., 1998]), ou permettant une seule substitution ([Tompa, 1999]), pour modéliser les sites à inférer, ce qui n'apporte en général pas suffisamment de souplesse pour les décrire. Pour pallier à cette faiblesse, d'autres algorithmes exacts infèrent des motifs sans erreurs sur des alphabets dégénérés ([Saurin and Marlière, 1987], [Neuwald and Green, 1994], [Jonassen, 1997], [Brazma et al., 1998b]). Plus récement, certains algorithmes se sont orientés vers l'inférence de paires de motifs séparés par une distance contrainte. Cette orientation est motivée par le fait que, chez certains organismes, la transcription est initiée par une association de plusieurs sites de fixation respectant des contraintes d'ordre et de distances plus ou moins fortes, conséquences d'interactions physico-chimiques complexes entre macromolécules (voir par exemple [Record et al., 1996], [Mulder et al., 1997], [Werner, 1999]). Cependant, aucune approche exacte autorisant des erreurs n'existe à ce jour pour inférer de telles associations de motifs.

Nous basant sur un algorithme exact récent inférant des motifs avec erreurs ([Sagot, 1998]), notre travail a consisté en premier lieu à étendre la notion de motif utilisée afin de prendre en compte ces contraintes d'ordre et de distance. L'enjeu d'une telle extension est la description plus fine des spécificités biologiques des sites de fixation, et le développement d'algorithmes apportant une sensibilité et une efficacité accrues. Nous avons ensuite introduit de nouveaux algorithmes exacts inférant de tels motifs étendus, et implanté ces algorithmes dans un outil générique.

Cette thèse est divisée en sept chapitres. Dans le premier, nous présentons le contexte du problème biologique auquel nous nous sommes intéressés. Nous introduisons d'abord les bases de biologie moléculaire nécessaires à sa compréhension. Puis, nous exposons le rôle particulier des sites de fixation dans la machinerie génétique et essayons de dégager leurs principales caractéristiques connues.

Dans le chapitre 2, nous faisons état des caractéristiques de ces sites qui peuvent être représentées et exploitées dans un modèle informatique. Nous parcourons les principales représentations utilisées à ce jour pour modéliser et manipuler *in silico*[1] des sites de fixation. Lors de cette transition, les premières difficultés se font jour concernant les compromis à faire dans le choix d'une telle représentation, privilégiant une description fine des spécificités des sites de fixation ou une efficacité de manipulation permettant de développer des algorithmes de détection performants.

Dans le chapitre 3, nous présentons le problème algorithmique auquel nous sommes

[1] sur ordinateur, par opposition aux expériences biologiques de laboratoire, *in vivo* ou *in vitro*.

confrontés, qui est une des projections possibles du problème biologique initial. Nous parcourons ensuite l'état de l'art dans le domaine de la détection de sites de fixation, et présentons rapidement certains algorithmes développés. Nous constatons dès lors les divergences entre deux types d'approches dans ce domaine, et les lacunes existantes concernant la représentation de certaines spécificités des sites de fixation. Au terme de ce chapitre, nous disposons des connaissances nécessaires pour motiver des choix précis en matière de représentations et d'algorithmes. Nous discutons donc des possibilités qui nous sont offertes et présentons les directions vers lesquelles nous avons orienté nos travaux de recherche.

Le chapitre 4 décrit les bases algorithmiques qui sont à l'origine de nos travaux. Le cahier des charges présenté au chapitre précédent leur est appliqué et nous détaillons les développements à y apporter.

Dans le chapitre 5, nous présentons la notion de *modèle structuré*, qui vise à décrire les capacités d'association de certains sites de fixation, et décrivons plusieurs nouveaux algorithmes inférant ces modèles. Une notion visant à réduire le parcours de l'espace des solutions d'un tel problème d'inférence tout en conservant un caractère exact à nos algorithmes est introduite et explorée.

Le chapitre 6 concerne l'implantation logicielle des algorithmes développés précédemment. Nous décrivons l'outil d'inférence de sites de fixation mis au point, nommé SMILE, et réalisons des expériences sur des données artificielles à l'aide de cet outil.

Enfin, nous fermons la boucle initiée par le problème biologique auquel nous nous sommes intéressés en appliquant, dans le dernier chapitre, l'outil que nous avons développé à l'inférence de sites de fixation dans les génomes de plusieurs organismes. Afin de valider la pertinence des choix algorithmiques que nous avons fait, nous comparons SMILE à certains outils d'inférence parmi les plus utilisés, en termes de sensibilité et d'efficacité. SMILE s'avère être à la hauteur de nos espérances, puisqu'il démontre dans certains cas une sensibilité supérieure à celle des outils existants, et réalise des inférences dans des séquences de tailles habituellement inexploitables de signaux très faiblement représentés, en des temps raisonnables.

La figure 2 présente un aperçu de l'organisation de cette thèse, mettant en avant les différents domaines traversés au fur et à mesure de ses chapitres. La diversité des compétences requises et des sujets abordés est particulièrement large dans le domaine de la bio-informatique en général, et dans nos travaux en particulier. À ce titre, nos recherches ont nécessité de nombreuses collaborations avec des chercheurs spécialisés en biologie moléculaire, statistiques, mathématiques discrètes et évaluation de probabilités. Notre apport est principalement algorithmique et, de ce point de vue, il nous faut insister sur le caractère générique des travaux que nous avons effectués. En effet, si leur motivation prend naissance dans un problème biologique précis, les structures, algorithmes et outils réalisés durant cette thèse sont à ce jour utilisables pour résoudre des problèmes similaires sur d'autres types de données, biologiques (des séquences protéiques par exemple) ou pas, et donc d'inférer des motifs structurés dans n'importe quelles conditions.

Au terme de ce mémoire, faisant état des résultats obtenus, nous observons le chemin parcouru et esquissons de nouvelles directions à suivre pour espérer détecter plus sensiblement et efficacement des sites qui restent indétectables à ce jour.

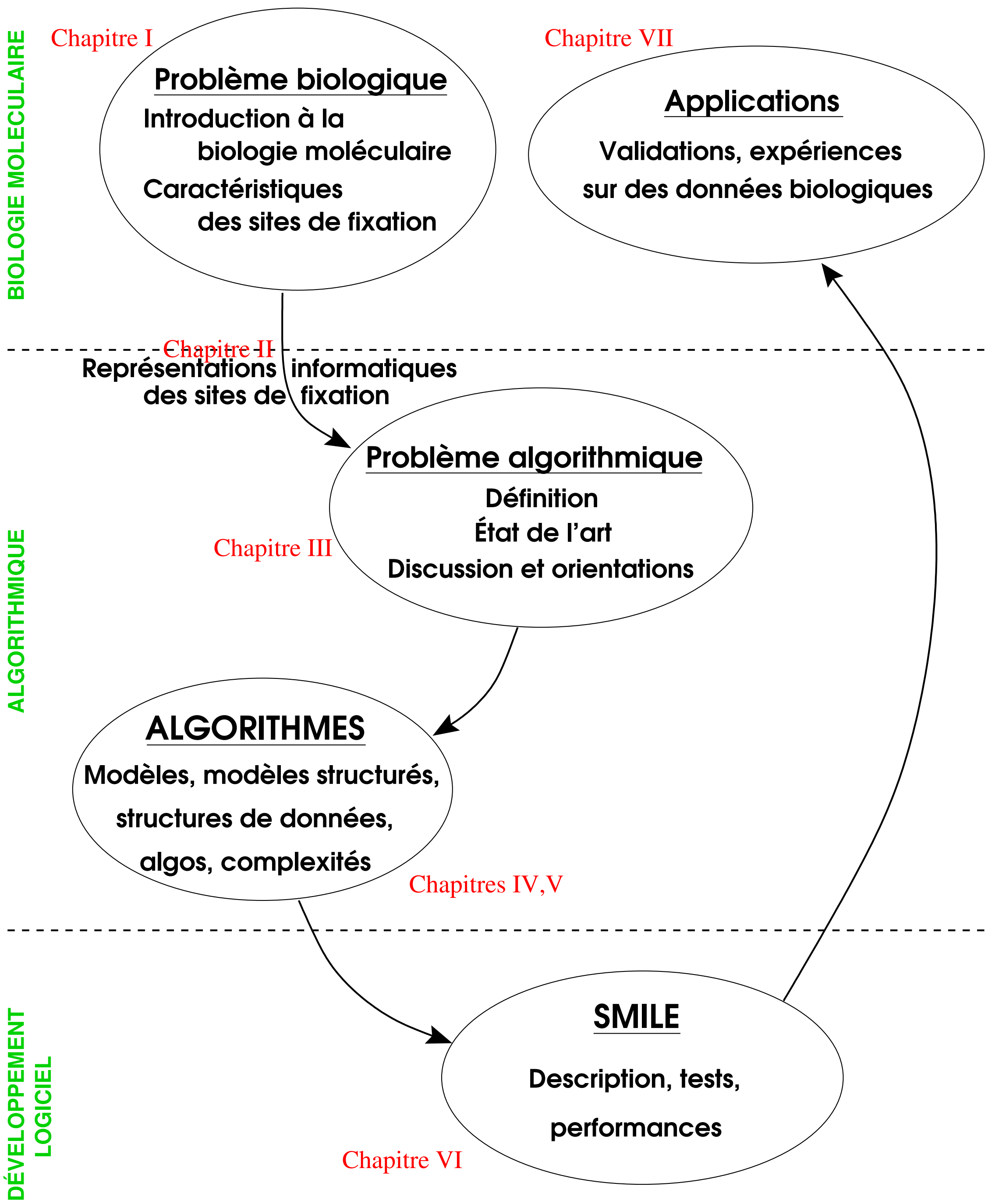

Fig. 2 – Plan de la thèse.

Chapitre 1

Une introduction au problème biologique

Vos enfants ne sont pas vos enfants.
Ils sont les fils et les filles de l'appel de la vie à elle-même.

Khalil Gibran.
Le prophète, 1923.

Nous introduisons dans ce chapitre les notions nécessaires à la compréhension du problème qui nous intéresse, la détection des sites de fixation dans les séquences d'ADN. Son objectif est donc de décrire les principales caractéristiques connues de ces sites afin de pouvoir en déduire des modèles informatiques pour les représenter. Nous invitons le lecteur souhaitant approfondir plus amplement ses connaissances dans le domaine de la biologie moléculaire à se référer aux nombreux ouvrages plus ou moins spécialisés qui existent, parmi lesquels nous nous contenterons de citer deux livres très complets ([Alberts et al., 1995] et [Lewin, 2000]) et, pour ceux que des ouvrages trop complexes découragent, un petit livre de vulgarisation des plus accessibles ([Jacquard, 1999]).

Délaissons pour le moment l'univers binaire de l'informatique pour nous plonger dans celui de la biologie moléculaire. Chaque terme biologique que nous emploierons pourrait à lui seul faire l'objet d'un chapitre, aussi doit-on garder à l'esprit que la simplicité avec laquelle nous décrivons ici certains mécanismes biologiques est tout à fait illusoire. De même, certains des points que nous présentons sont toujours très discutés à ce jour, et ne doivent donc pas être pris pour des vérités établies.

1.1 Les acteurs fondamentaux du vivant

Comment dissocier le vivant de l'inerte ? Cette question a suscité (et suscite toujours) de nombreuses controverses. En biologie, certains auteurs s'accordent à dire que, dans la

« soupe primordiale » qui recouvrait la surface de la Terre il y a 3 à 4 milliards d'années, parmi les myriades d'associations de molécules qui se font et se défont au gré des affinités physico-chimiques et des contraintes du milieu, l'entité qui doit être qualifiée de vivante est celle qui possède une propriété inédite et unique : la capacité de se reproduire, c'est-à-dire de s'opposer aux effets du temps. Cette macro-molécule est l'acide désoxyribonucléique, ou ADN [1].

1.1.1 L'ADN

L'ADN est le point commun de la quasi-totalité des être vivants sur Terre : les organismes simples (comme les bactéries) ou complexes (comme l'homme) contiennent une ou plusieurs macro-molécules d'ADN qui constituent son *matériel génétique*, l'information nécessaire à sa construction et à son fonctionnement. Ce matériel génétique est contenu dans toutes les cellules d'un organisme. On distingue deux grandes familles de cellules : les procaryotes, sans noyau, et les eucaryotes, où l'ADN est détenu dans un noyau.

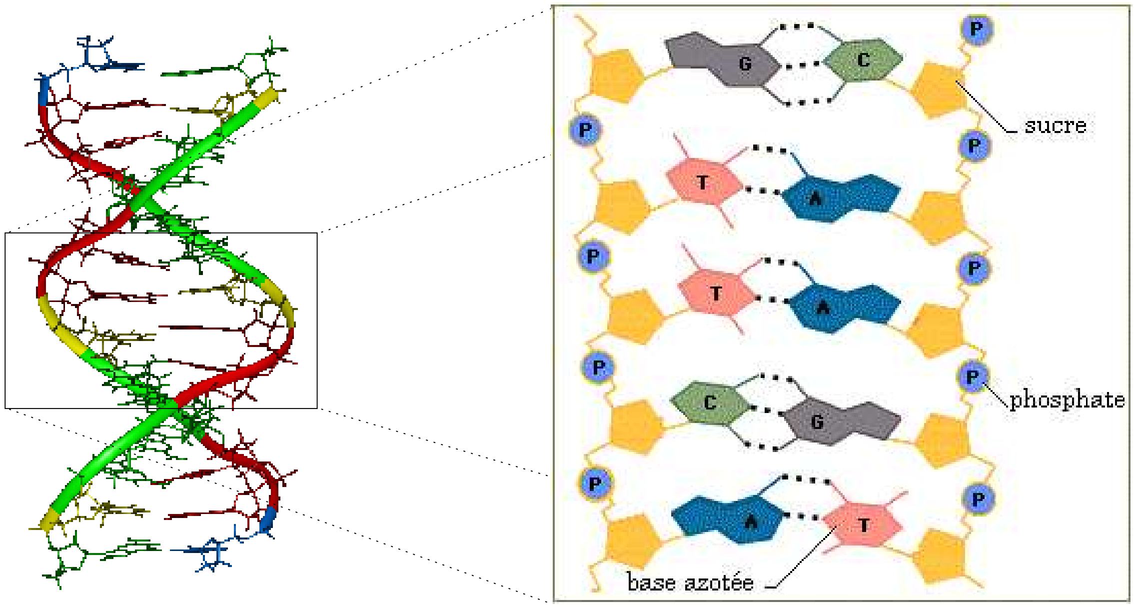

FIG. 1.1 – La structure de l'ADN.©Rockfeller University et Lycée Montesquieu (95).

La molécule d'ADN est constituée de deux brins antiparallèles orientés formant une double hélice (voir la figure 1.1). Chacun des deux brins d'ADN est un enchaînement de *nucléotides*, constitués :

– d'un sucre et d'un groupement phosphate ;

[1] certaines théories des origines de la vie avancent l'idée que la molécule d'ARN aurait précédé l'ADN.

– d'une base azotée.

La succession de sucres et de phosphates est répétitive : elle ne contient pas d'information, et joue un rôle de squelette dans la structure de l'ADN. Ce sont les bases azotées qui expriment la diversité du brin d'ADN. Quatre bases azotées différentes peuvent en effet composer un nucléotide : l'adénine (notée A), la cytosine (C), la guanine (G), et la thymine (T). Dans une molécule d'ADN, les deux brins parallèles sont reliés entre eux par les liaisons hydrogènes qui réunissent leurs bases azotées (en pointillés sur la figure 1.1). Pour des raisons stéréochimiques, une adénine est toujours associée à une thymine, et une cytosine à une guanine. Les deux brins d'une molécule d'ADN sont donc complémentaires : chaque brin peut être déduit de la composition de l'autre.

La molécule d'ADN prend, lors de certaines phases du fonctionnement de la cellule, une conformation « comprimée » en s'enroulant autour de protéines (les *histones*) pour former un *chromosome*. Certains organismes plus complexes possèdent plusieurs chromosomes (deux fois 23 chez l'homme). Le terme *génome* désigne les successions de bases contenues dans l'ensemble des brins d'ADN d'un organisme. Le génome de l'homme contient 3,3 milliards de bases.

1.1.2 L'ARN

L'acide ribonucléique ou ARN est aussi une macro-molécule constituée d'une succession de nucléotides, formant cette fois un simple brin. Les nucléotides employés sont les mêmes que ceux de l'ADN, à l'exception de la thymine qui est remplacée par l'uracile (U) mais s'apparie toujours avec l'adénine. Il existe plusieurs types d'ARN qui jouent pour la plupart un rôle d'intermédiaire dans la synthèse des protéines à partir de l'ADN.

Les molécules d'ARN sont beaucoup plus courtes que les molécules d'ADN, de l'ordre de quelques milliers de nucléotides. Leur structure en simple brin n'est pas aussi régulière, et beaucoup moins stable, que la double hélice de l'ADN. Les molécules d'ARN sont sujettes à des repliements tri-dimensionnels complexes, conséquences d'appariements entre différents nucléotides d'un même brin.

La table des symboles IUPAC (tableau 1.1) permet de décrire les nucléotides qui composent une séquence d'ADN ou d'ARN.

1.1.3 Les protéines

Les protéines sont les molécules fonctionnelles des processus biologiques, actrices principales du fonctionnement de la cellule. Elles jouent des rôles très variés au sein de cette cellule, répartis en deux grands groupes :

– les protéines membranaires, qui font partie des constituants de la membrane cellulaire. Elles jouent à ce titre un rôle primordial dans l'équilibre du milieu intra-cellulaire, et dans le passage de molécules de la cellule au milieu extérieur (et inversement) ;
– les protéines globulaires, qui comprennent des protéines aux fonctionnalités variées. Certaines d'entre elles servent de catalyseurs à des réactions du métabolisme de la

Symbole	Signification	Commentaire
A		Adénine
C		Cytosine
G		Guanine
T		Thymine
U		Uracyle, équivalent de T dans les séquences d'ARN
R	A, G	Purine
Y	C, T	Pyrimidine
W	A, T	Faible (weak) interaction de la liaison hydrogène
S	C, G	Forte (strong) interaction de la liaison hydrogène
M	A, C	Groupes NH_2 (amino) en positions identiques
K	G, T	Groupes ketones en positions identiques
B	C, G, T	Tout sauf A (premier symbole libre qui suit A)
D	A, G, T	Tout sauf C (premier symbole libre qui suit C)
H	A, C, T	Tout sauf G (premier symbole libre qui suit G)
V	A, C, G	Tout sauf T (premier symbole libre qui suit T)
N	A, C, G, T	N'importe quel (any) symbole

TAB. 1.1 – Alphabet IUPAC des nucléotides ([Moss, 2001]).

cellule (les *enzymes*). D'autres ont un rôle de médiateur entre différentes cellules d'un même organisme (les *hormones*) ou de défense vis-à-vis des corps étrangers à l'organisme (les *anticorps*).

Les protéines sont des assemblages de molécules appelées *acides aminés*. Une protéine est constituée d'une succession de quelques dizaines à quelques milliers d'acides aminés. Ces acides aminés sont au nombre de vingt, et possèdent des propriétés physico-chimiques très variées (présentées dans le tableau 1.2 sous leur forme IUPAC).

Du fait des interactions entre acides aminés dues à ces propriétés physico-chimiques, les protéines subissent des repliements qui leur font adopter une structure tri-dimensionnelle complexe. Cette structure est d'une grande importance car c'est d'elle que dépendent les interactions qu'une protéine va créer avec d'autres molécules. La structure d'une protéine explique également probablement sa fonction.

Acide aminé	Code à trois lettres	Code à une lettre	Propriétés physico-chimiques
Alanine	Ala	A	très petit, hydrophobe
Argine	Arg	R	positif, polaire
Asparagine	Asn	N	petit, polaire
Acide Aspartique	Asp	D	petit, négatif
Cystéine	Cys	C	petit, polaire, hydrophobe
Acide Glutamique	Glu	E	polaire, négatif
Glutamine	Gln	Q	polaire
Glycine	Gly	G	très petit, hydrophobe ou hydrophile
Histidine	His	H	aromatique, hydrophobe, positif, polaire
Isoleucine	Ile	I	aliphatique, hydrophobe
Leucine	Leu	L	aliphatique, hydrophobe
Lysine	Lys	K	hydrophobe, polaire, positif
Méthionine	Met	M	hydrophobe
Phénylalanine	Phe	F	aromatique, hydrophobe
Proline	Pro	P	petit
Sérine	Ser	S	très petit, polaire
Thréonine	Thr	T	petit, hydrophobe, polaire
Tryptophane	Trp	W	aromatique, hydrophobe, polaire
Tyrosine	Tyr	Y	aromatique, hydrophobe, polaire
Valine	Val	V	petit, aliphatique, hydrophobe

TAB. 1.2 – Alphabet IUPAC des acides aminés, codes et principales propriétés physico-chimiques.

1.2 Les mécanismes fondamentaux du vivant

Munis de ces brèves descriptions des trois principales molécules du vivant, l'ADN, l'ARN et les protéines, nous présentons dans cette partie les principaux mécanismes mis en jeu dans une cellule, et dont elles sont les actrices majeures. Ces mécanismes visent deux objectifs primordiaux : reproduire le matériel génétique de la cellule, et assurer son fonctionnement.

1.2.1 La réplication de l'ADN

Nous avons vu précédemment qu'une des propriétés de l'ADN était de posséder deux brins complémentaires, tels qu'on peut reconstruire la molécule complète à l'identique à partir d'un des deux brins. C'est cette propriété fondamentale de complémentarité qui donne son pouvoir de reproduction à l'ADN. Quand l'ADN est dans une phase de *réplication*, par le biais de nombreux mécanismes complexes, les deux brins d'une molécule d'ADN sont dissociés, et de nouveaux nucléotides sont associés à chacun d'eux, recomposant ainsi le brin manquant par complémentarité. Au terme de ce processus, on dispose de deux molécules d'ADN identiques à la molécule initiale (voir figure 1.2).

1.2.2 La synthèse protéique

Les protéines sont les principales protagonistes du fonctionnement de la cellule. L'ADN d'une cellule contient la description :

- des « plans de construction » des protéines nécessaires à la cellule ;
- des rythmes de fabrication de ces protéines en fonction des besoins de la cellule.

Nous décrivons ici, très superficiellement toujours, les mécanismes qui prennent place dans la synthèse des protéines à partir de l'information contenue dans l'ADN.

1.2.2.1 L'organisation de l'information dans l'ADN

L'ADN, support de l'information génétique, utilise un alphabet à quatre symboles (A, C, G et T). Cet alphabet restreint permet cependant d'exprimer toutes les informations nécessaires au fonctionnement de la cellule, de même que les deux symboles du Morse permettent de décrire, laborieusement certes, n'importe quel texte écrit avec des caractères latins. On a longtemps pensé que les protéines, exprimées sur un alphabet plus riche, détenaient l'information génétique, l'ADN ayant été jugé de composition trop simple pour cela.

Toutes les zones d'un brin d'ADN ne sont pas fonctionnelles : on distingue les *gènes*, qui contiennent les « plans de construction » des ARN et protéines dont a besoin la cellule, du reste d'une séquence d'ADN, appelé *non-codant* ou non-gène, qui n'est probablement pas fonctionnel en grande partie (le rôle du non-codant reste assez obscur en l'état actuel des connaissances) mais contient toutefois les signaux indispensables à la régulation de la synthèse protéique. Nous parlerons de ces signaux plus loin. Chez l'homme, les gènes,

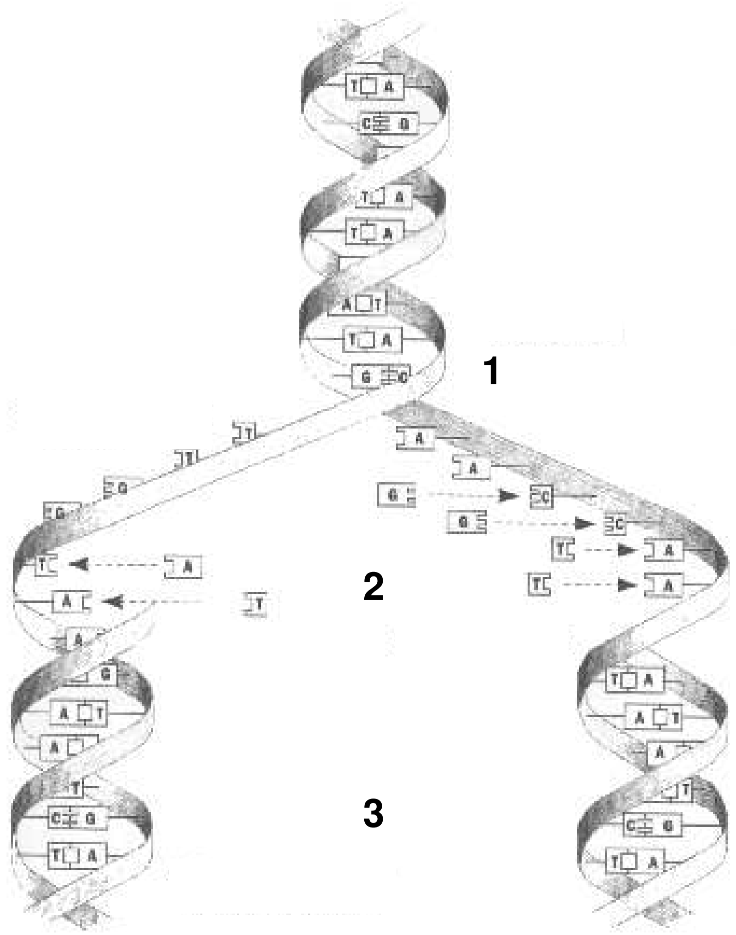

FIG. 1.2 – La réplication de l'ADN.
Tiré de library.thinkquest.org. 1) Les deux brins sont dissociés, 2) des nucléotides libres viennent compléter chacun des deux brins, 3) deux molécules identiques à la molécule initiale sont finalement construites.

au nombre de 30 à 80 000 ([Hollon, 2001]), représentent 5% des 3,3 milliards de bases qui constituent son génome. Chez les organismes procaryotes, l'ADN, presque toujours circulaire, contient en revanche très peu de non-codant.

Les deux brins d'une molécule d'ADN possèdent un sens de lecture opposé, les gènes se trouvant sur un brin sont tous lus dans la même direction. Deux gènes peuvent se trouver face à face sur chacun des deux brins, ils seront lus dans des directions différentes.

La synthèse protéique consiste en l'interprétation d'un gène de l'ADN en une succession d'acides aminés qui s'associeront pour former une protéine. La première étape de cette interprétation est la transcription de l'ADN en ARNm.

1.2.2.2 La transcription de l'ADN en ARNm

La transcription désigne la fabrication d'un ARNm (messager) à partir d'un gène d'un des brins de l'ADN. Comme lors de la réplication, ce processus a lieu par complémentation d'un brin. Les différences majeures avec la réplication sont que seule une partie du brin d'ADN (celle contenant le gène) est utilisée, et que le brin synthétisé est un ARNm, qui se dissocie du brin d'ADN ayant servi à sa construction. Au terme de la transcription, l'ADN, dont les deux brins ont momentanément été séparés sur une courte longueur, retrouve son état initial.

Chez les procaryotes, l'ARNm ainsi synthétisé est souvent *mature* (il peut dans certains cas subir ensuite des altérations à ses extrémités), c'est-à-dire qu'il est fonctionnel, prêt à être utilisé pour synthétiser une protéine par exemple. Chez les eucaryotes, le procédé de synthèse d'un ARNm mature est plus complexe. Les gènes des eucaryotes sont en effet constitués de zones codantes (les *exons*) et non-codantes (les *introns*, dont le rôle est mal connu). L'ARNm synthétisé doit être découpé de façon à ce que les introns soient ôtés. Au terme de cette phase nommée *épissage* (et d'autres éventuelles modifications sur l'ARNm transcrit), l'ARNm est mature.

1.2.2.3 La traduction de l'ARNm en protéine

Les deux symboles du langage Morse doivent être assemblés en différentes combinaisons pour représenter, par exemple, l'un des 26 symboles de notre alphabet français. De la même manière, les nucléotides, exprimés sur un alphabet à 4 symboles, doivent être combinés pour exprimer l'un des 20 acides aminés. La combinaison de deux nucléotides ne peut fournir que $4^2 = 16$ couples différents. La représentation des 20 acides aminés nécessite donc de combiner au moins trois nucléotides. Le passage de l'alphabet des nucléotides à celui des acides aminés se fait effectivement par la lecture de triplets de nucléotides dénommés *codons*. Sur l'alphabet de l'ADN (ou de l'ARNm), on peut écrire $4^3 = 64$ codons différents. Le code qui permet de passer des codons aux acides aminés est donc dégénéré : plusieurs codons expriment le même acide aminé (voir la table 1.3). Ce code génétique est, à quelques exceptions près, universel à tous les êtres vivants. Parmi les 64 codons possibles, 61 codent pour des acides aminés. Les codons UAA, UAG et UGA sont dits « codons-stop », et indiquent la fin de la traduction.

La *traduction* est l'interprétation des codons de l'ARNm en acide aminé. Les principaux acteurs de cette étape de la synthèse protéique sont le *ribosome* (constitué de protéines et de plusieurs types d'ARNr - ribosomal), et les ARNt (de transfert). Notons qu'un gène code pour une chaîne polypeptidique, et qu'une protéine peut être constituée de plusieurs de ces chaînes.

Les processus de transcription et de traduction constituent les étapes du dogme central de la biologie cellulaire, qu'on schématise par :

$$\textbf{ADN} \rightarrow \textbf{ARN} \rightarrow \textbf{Protéines}$$

1ère base	2ème base				3ème base
	U	**C**	**A**	**G**	
U	Phe(F)	Ser(S)	Tyr(Y)	Cys(C)	**U**
	Phe(F)	Ser(S)	Tyr(Y)	Cys(C)	**C**
	Leu(L)	Ser(S)	**STOP**	**STOP**	**A**
	Leu(L)	Ser(S)	**STOP**	Trp(W)	**G**
C	Leu(L)	Pro(P)	His(H)	Arg(R)	**U**
	Leu(L)	Pro(P)	His(H)	Arg(R)	**C**
	Leu(L)	Pro(P)	Gln(Q)	Arg(R)	**A**
	Leu(L)	Pro(P)	Gln(Q)	Arg(R)	**G**
A	Ile(I)	Thr(T)	Asn(N)	Ser(S)	**U**
	Ile(I)	Thr(T)	Asn(N)	Ser(S)	**C**
	Ile(I)	Thr(T)	Lys(K)	Arg(R)	**A**
	Met(M)	Thr(T)	Lys(K)	Arg(R)	**G**
G	Val(V)	Ala(A)	Asp(D)	Gly(G)	**U**
	Val(V)	Ala(A)	Asp(D)	Gly(G)	**C**
	Val(V)	Ala(A)	Asp(D)	Gly(G)	**A**
	Val(V)	Ala(A)	Asp(D)	Gly(G)	**G**

TAB. 1.3 – Le code génétique.
À chaque codon correspond un acide aminé. Notons que les variations sur la troisième base d'un codon n'entraînent pas toujours de changement d'acide aminé. En particulier, U et C sont indifférenciés en dernière position. Chaque acide aminé est codé par 1 à 6 codons.

Mais cette machinerie, décrite ici de manière très simpliste, ne fonctionne pas de façon régulière : la cellule subit en permanence des variations dans son milieu, à cause de facteurs intra ou extra-cellulaires. Chaque protéine, qui possède un ou plusieurs rôles précis, est produite en fonction du contexte dans lequel se trouve la cellule, de ses besoins et de son état.

L'aperçu rapide des mécanismes fondamentaux de la vie présenté dans cette section nous permet de nous poser certaines questions élémentaires en rapport avec le fonctionnement de la cellule, qui vont nous rapprocher petit à petit de la problématique à laquelle nous nous sommes intéressés.

La question qui débute cette démarche est la suivante : comment la cellule procède-t-elle pour réguler toute cette machinerie ? Pour comprendre un peu mieux les processus qui permettent de contrôler la synthèse protéique, il nous faut décrire plus précisément certains mécanismes. Si ce contrôle semble être exercé tout au long du processus de synthèse protéique, nous nous intéressons plus particulièrement à l'étape de transcription. L'initiation de la transcription est en effet une des étapes majeures dans la régulation des gènes. Le fait qu'un gène soit transcrit est le point de départ de la synthèse de la protéine associée à ce gène. Une faible partie des ARNm synthétisés est cependant traduite : certains sont directement fonctionnels, d'autres sont dégradés avant de pouvoir être traduits, ce qui constitue une autre régulation, dite post-transcriptionnelle. Nous dirons qu'un gène est *exprimé* quand il atteint son but : un gène codant pour une protéine est exprimé quand la protéine est synthétisée.

1.2.3 Une étape importante dans la régulation des gènes : l'initiation de la transcription

Nous avons décrit précédemment la transcription comme l'étape qui permet la transformation de l'ADN en ARNm. Parmi les nombreux détails de ce processus que nous avons éludés, nous souhaitons éclaircir deux points :
- quelle molécule déclenche cette transcription ?
- comment le début d'un gène est-il repéré par cette molécule le long de l'ADN ?

1.2.3.1 L'ARN polymérase

La synthèse de l'ARNm résulte de l'action d'un enzyme, l'*ARN polymérase*. Cet enzyme vient se fixer sur une zone particulière de l'ADN, la *région promotrice*, qui indique où doit débuter la transcription du gène qui se trouve en aval (dans le sens de lecture du brin concerné). La fixation de l'ARN polymérase sur la région promotrice entraîne la séparation localisée des deux brins d'ADN. L'ARN polymérase se déplace ensuite sur le brin à transcrire, comme une tête de lecture, et synthétise l'ARNm complémentaire de l'ADN lu. Par convention, le premier nucléotide transcrit est numéroté +1, et ceux qui le suivent +2, +3... En amont de la zone transcrite, on numérote également les nucléotides en fonction de ce premier nucléotide transcrit : -1, -2, -3...

1.2.3.2 Cas des procaryotes

Chez les procaryotes, un seul type d'ARN polymérase, l'ARN pol, assure la transcription. Il est composé de quatre sous-unités (ou *facteurs*) dont l'un, le facteur σ, lui permet de reconnaître des signaux se trouvant dans la région promotrice (aussi appelée promoteur). Ces signaux sont des séquences nucléiques particulières sur lesquelles le facteur de transcription σ vient se fixer : ce sont des *sites de fixation*.

Les promoteurs

Les promoteurs bactériens connus contiennent en général deux sites de fixation, aux environs des positions -10 et -35, et constitués d'environ six nucléotides. Des légères variations dans la composition ou la position d'un site de fixation peuvent influer sur la fréquence d'expression d'un gène, et donc sur la quantité de protéines correspondantes synthétisées.

Nous avons vu que l'expression des gènes est dépendante du contexte dans lequel se trouve la cellule. Chez *E.coli* par exemple, il existe un mécanisme connu sous le nom de

« réponse SOS », qui est déclenché en réponse à une grave lésion de l'ADN. Tout blocage de la réplication de l'ADN produit un signal qui induit une augmentation de la transcription de plus de quinze gènes, dont beaucoup codent des protéines qui jouent un rôle dans la réparation de l'ADN ([Alberts et al., 1995]). Dans un certain contexte, un groupe de gènes impliqués dans la synthèse de protéines ayant une action particulière peut être plus fortement exprimé. On dira que ces gènes sont *co-exprimés*. Un même facteur de transcription peut réguler plusieurs gènes : on dit alors qu'ils sont *co-régulés* transcriptionnellement. Certains gènes consécutifs sont régis par un unique promoteur, qui initie à lui seul leur transcription : ces *gènes co-transcrits* forment alors un *opéron*. Les protéines codées par les gènes d'un opéron participent souvent à une même chaîne de réactions métaboliques.

Sites de régulation et facteurs de transcription

Des facteurs de transcription supplémentaires s'associent parfois à l'ARN polymérase, pouvant créer de nouvelles interactions avec d'autres sites de fixation (figure 1.3).

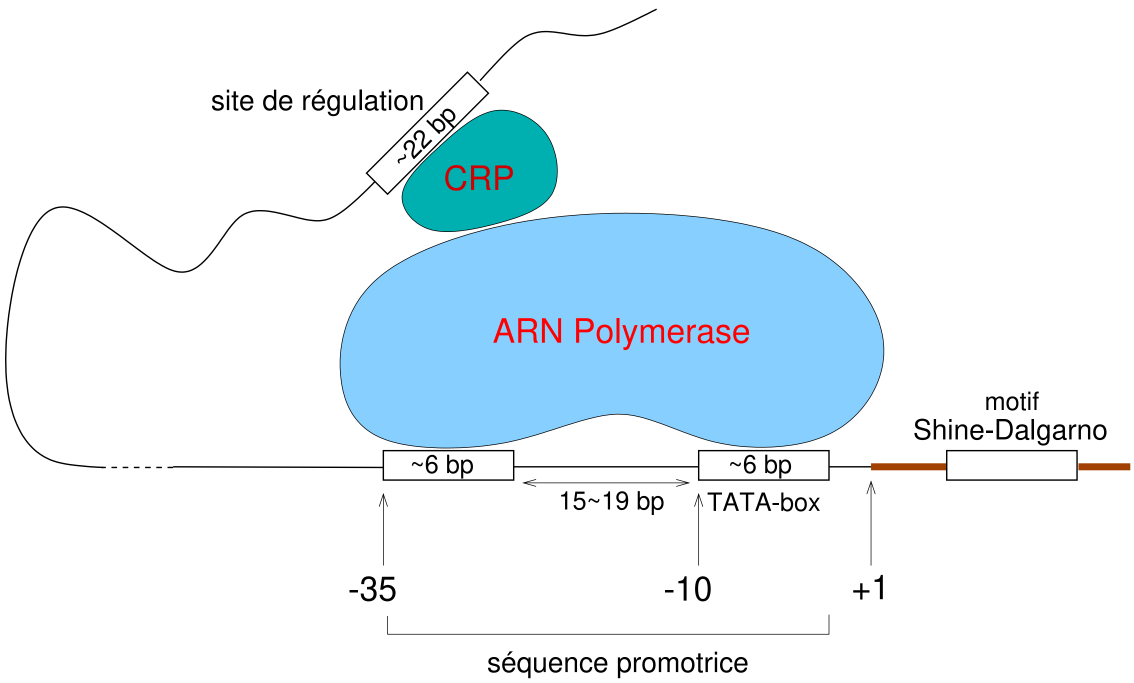

Fig. 1.3 – Un schéma simplifié des interactions entre l'ARN polymérase, la région promotrice et un site de régulation d'un gène chez *E.coli*.
Les deux sites de fixation (dont l'un est bien connu sous le nom de *TATA-box*) sont en général distants de 15 à 19 bases. Le facteur CRP vient se fixer sur l'ARN polymérase pour interagir avec un site de régulation qui peut se trouver parfois loin en amont du début de la transcription. Le motif Shine-Dalgarno est un signal, au même titre que les sites d'initiation de la transcription, qui intervient comme initiateur de la traduction.

Ces interactions peuvent avoir un effet *activateur* ou *inhibiteur* sur l'initiation de la transcription. La présence de tels sites de fixation (appelés *sites de régulation*) au bon endroit en amont d'un gène ou dans ce même gène peut donc avoir un effet régulateur sur la transcription. Les sites de régulation se situent parfois loin du gène sur lequel ils ont une influence.

Encore une fois, la nature des interactions que va générer un facteur de transcription avec le brin d'ADN dépendent fortement de sa structure tri-dimensionnelle (voir par exemple la figure 1.4).

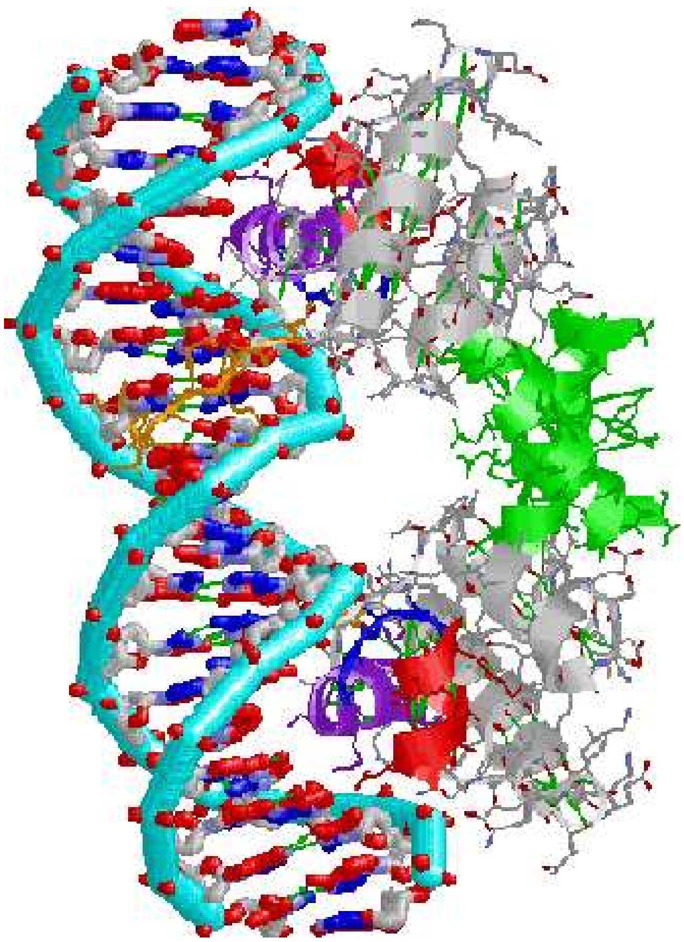

FIG. 1.4 – Interaction (en deux points) du répresseur λ du bactériophage Lambda avec la molécule d'ADN. Protein Data Base ID :1LMB. (Beamer et Pabo, 93).

1.2.3.3 Cas des eucaryotes

Sur son principe, l'initiation de la transcription chez les eucaryotes fonctionne comme celle des procaryotes. Les interactions mises en jeu y sont cependant plus complexes. Il existe trois types d'ARN polymérases : les ARN pol I, pol II et pol III, qui synthétisent respectivement tous les ARNr (ribosomaux) longs, l'ARNm, et les ARNt et ARNr courts. Il n'y a pas d'opérons, chaque ARNm code donc pour une seule chaîne polypeptidique. Les sites de régulation sont parfois très nombreux et très en amont du gène régulé (figure 1.5). Les facteurs interagissent coopérativement et peuvent se fixer en plusieurs points du brin d'ADN (voir [Werner, 1999], et [Sinha, 2001]). On connaît dans certains cas plus d'une vingtaine de sites de fixation impliqués dans la régulation de la transcription d'un gène ([Arnone and Davidson, 1997], [Masternak et al., 2000]). L'ordre dans lequel se trouvent ces sites et les distances qui les séparent sont d'une importance très variable. Dans certains cas, la moindre variation dans ces caractéristiques n'est pas tolérée ([Linhoff et al., 1997]). Dans d'autres cas, des interversions dans les positions des sites, des délétions de sites complets ou des modifications importantes dans les distances qui les séparent, bien qu'ayant une influence sur le taux d'expression du gène associé, n'empêchent pas l'initiation de la transcription ([Ohmori et al., 1997]). Les distances entre sites reconnus par un même facteur, généralement réduites, peuvent parfois dépasser 150 bases. Il semble que la classe de structures à laquelle appartient un facteur de transcription puisse avoir une influence sur la distance à laquelle ce facteur interagit avec d'autres facteurs ([Sinha, 2001]). Il arrive qu'un facteur interagisse avec un site se trouvant en aval du début de la transcription. Enfin, la rigidité des contraintes de distance et d'ordonnancement semble propre aux facteurs concernés : lors d'une même initiation, certains sites doivent parfois respecter des contraintes très strictes tandis que d'autres n'en subissent quasiment pas.

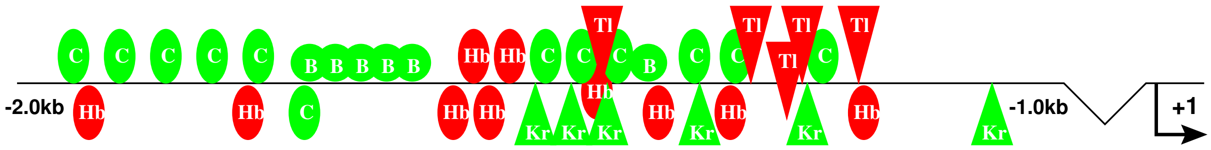

FIG. 1.5 – Éléments *cis*-régulateurs du gène *kni* chez *Drosophila*.
Le trait noir figure le brin d'ADN, et la flèche notée +1 indique le début et le sens de la transcription. Les sites de régulation présentés se situent à plus de 1000 bases en amont du gène. Les répresseurs sont en rouge (foncé) et les activateurs en vert (clair). Source [Arnone and Davidson, 1997].

Malgré cette diversité importante, certains éléments sont communs aux différents mécanismes d'initiation de la transcription. On retrouve par exemple fréquemment le classique facteur TBP (TATA *binding factor)* qui reconnaît un site de fixation nommé *TATA-box*. Les séquences promotrices se trouvent en général autour des positions -25 et -75.

1.2.4 Les mutations

Les mécanismes fondamentaux de la cellule sont sujets à des erreurs, appelées *mutations*. Elles peuvent survenir, entre autres, lors de la réplication, quand les brins d'ADN complémentaires sont synthétisés, se manifestant par :

- une *substitution* d'une base par une autre base ;
- une *insertion* d'une base ;
- une *délétion* d'une base.

Les mutations qui ont lieu lors de cette étape sont d'une grande importance car, pour peu qu'elles n'interfèrent pas avec la capacité de reproduction de la cellule, elles seront transmises aux générations suivantes. Les mutations qui apparaissent lors de la transcription ont par exemple moins d'importance, puisqu'elles n'impliquent la synthèse que d'un seul ARNm (et éventuellement d'une protéine) modifié.

1.2.4.1 L'influence des mutations ayant lieu lors de la réplication

L'influence des mutations ayant lieu lors de la réplication dépend des régions du génome où elles ont lieu. Dans des zones codantes d'un gène, une mutation peut être fatale au bon fonctionnement de l'intégralité de la cellule : nous avons vu que les nucléotides étaient lus par codons pour être traduits en acides aminés. Une simple substitution aura pour plus grande conséquence de modifier un des acides aminés de la chaîne polypeptidique produite (ce qui peut tout de même rendre une protéine inopérante). Mais une insertion ou une délétion dans une région codante entraîne un décalage dans la lecture des codons (on parle de *phase de lecture*), qui modifie intégralement la chaîne polypeptidique codée. Pour peu que ce gène code pour une protéine ayant un rôle majeur dans le fonctionnement de la cellule, cette simple mutation aura bouleversé ce fonctionnement.

Intervenant en dehors des gènes, une mutation peut avoir une influence nulle si aucun site de fixation utile n'est concerné et qu'elle n'implique pas de modification cruciale de la conformation du brin d'ADN. Dans le cas contraire, elle peut modifier la régulation de ce gène, accroissant ou diminuant son expression.

Les mutations peuvent avoir des conséquences très variable sur le fonctionnement de la cellule, améliorant ses performances ou entraînant sa mort. Elles ont donc un rôle important dans l'évolution de l'organisme. De nombreux mécanismes de correction existent qui vérifient l'exactitude des copies réalisées lors de la réplication, ou, comme dans l'exemple de la réponse SOS de *E.coli*, interviennent *a posteriori*, lorsque les effets de cette mutation ont un impact sur le fonctionnement. Certains travaux ([Lewin, 2000]) ont montré que les mutations n'interviennent pas au même rythme suivant leur position à l'intérieur du génome. Les zones codantes sont, par exemple, moins touchées que les autres. De même, à l'intérieur des zones non-codantes, les sites de fixation, qui ont un rôle d'importance dans le fonctionnement de toute la machinerie cellulaire, sont supposés être mieux conservés que le reste du non-codant. Ces observations sont probablement les conséquences de la sélection qui s'opère sur les organismes dont le métabolisme est modifié.

1.2.4.2 Variabilité des sites de fixation

Certains facteurs de transcription ont des affinités avec des régions promotrice particulières, et initient ainsi la transcription. Les mutations subies par ces régions ont un lien direct avec la régulation du gène concerné. En effet, un facteur de transcription qui a par exemple une affinité importante avec la séquence TTGACA pourra également s'associer avec une séquence proche, comme TTAACA, ou TTGGACA. Lors d'expériences en laboratoire, on observe souvent qu'un facteur de transcription donné a tendance à s'associer avec des séquences ayant certains points communs. On en déduit un *consensus*, un patron de la séquence reconnue par ce facteur. L'affinité de ce facteur de transcription avec des variantes du consensus étant moindre, il se peut que la transcription d'un gène dont le promoteur a subi des mutations soit initiée moins fréquemment qu'elle ne l'est quand ils sont proches du consensus reconnu par le facteur. Des promoteurs dégénérés contribuent donc, en impliquant une plus faible fréquence de transcription, à réguler l'expression d'un gène.

De même que la composition d'un promoteur a une influence sur son activité, la position des sites de fixation qui le composent est également importante. Un facteur de transcription qui, comme c'est souvent le cas, reconnaît deux sites de fixation distants d'un certain nombre de bases, peut, en fonction de sa structure, permettre une certaine variabilité de cette distance. Des interactions entre plusieurs facteurs de transcriptions associés peuvent aussi induire des contraintes de distance entre les sites qu'ils reconnaissent. Par exemple, le consensus du facteur σ^{70} observé chez *E.coli* ([Record et al., 1996]) est composé de deux sites de fixation, TTGACA et TATAAT, séparés par 15 à 19 bases. Les propriétés structurales de l'ADN font qu'un tour complet de la double hélice est réalisé environ toutes les dix bases. La distance entre ces deux sites implique donc une certaine conformation du facteur de transcription qui les reconnaît. De plus, la structure de l'ADN forme des petits et grands sillons le long de la double hélice, les grands sillons étant plus propices aux interactions avec une structure extérieure. Ces deux propriétés structurales ne sont que des exemples de la diversité des interactions physico-chimiques qui sont probablement prises en compte lors de la reconnaissance d'une région promotrice.

1.3 La détection des sites de fixation

1.3.1 Le rôle de la bio-informatique

Cette introduction biologique, résolument orientée vers la description des sites de fixation, introduit le sujet auquel nous nous sommes intéressés : la détection informatique de ces séquences nucléiques spécifiques. Les sites de fixation qui sont connus à ce jour pour certains organismes ont été déterminés expérimentalement, par des manipulations et observations complexes, souvent très longues et coûteuses. L'accroissement exponentiel des données génétiques disponibles a amené les bio-informaticiens à s'interroger sur les propriétés de ces sites qui permettraient de les détecter à moindre coût et à plus grande échelle, de façon automatisée, à partir des données disponibles dans les banques. Faute de connaître les caractéristiques exactes qui font qu'une séquence nucléique est un site de fixation pour un facteur de transcription, nous nous interrogeons donc sur les propriétés connues et supposées que nous pourrions exploiter. Ces travaux de détection informatique n'ont pas pour ambition de se substituer aux expérimentations biologiques, mais d'appréhender le problème de la détection des sites de fixation sous un angle nouveau, qui permettra peut-être découvrir certaines propriétés de ces sites. Les résultats obtenus avec de telles approches ne doivent pas être perçus comme des preuves expérimentales de la validité d'un site, mais comme des pistes permettant aux biologistes de faciliter la détermination d'un site.

1.3.2 Propriétés de sites de fixation

Il est difficile de dégager des propriétés générales concernant les sites de fixation dans leur ensemble, mais nous pouvons cependant d'ores et déjà constater certains points communs, et formuler certaines hypothèses.

Le problème de la détection de sites de fixation amène naturellement à se demander comment un facteur de transcription effectue cette reconnaissance. Si nous savons que les affinités qui se créent entre un site de fixation et un facteur de transcription sont dues à des propriétés physico-chimiques, nous ne savons pas, à ce jour, déduire de la structure ou de la composition d'un facteur de transcription, quand bien même celles-ci seraient connues précisément (une étude récente présente les caractéristiques de certains facteurs [Garvie and Wolberger, 2001] et de leurs interactions avec l'ADN), les sites auxquels il pourrait potentiellement se fixer. Il nous faut donc faire état des propriétés propres aux sites de fixation, en les considérant, autant que possible, indépendamment des facteurs de transcription.

Caractère « exceptionnel »

Il peut tout d'abord sembler logique d'imaginer qu'un site de fixation, qui possède certaines propriétés déclenchant des processus complexes, n'apparaisse pas n'importe où dans une séquence d'ADN. Si ce site est présent dans des zones du génome où il ne régule aucune transcription, il pourrait provoquer des fixations inutiles et coûteuses en énergie pour la cellule. Cette supposition mérite cependant d'être fortement atténuée, pour au moins deux raisons :

- en l'état actuel des connaissances, nous ne savons pas quelle est la « vision » qu'un facteur de transcription possède sur un génome. Il est possible que certaines zones ne lui soient pas accessibles, et que, par conséquent, elle ne représentent pas des zones potentielles de transcription, même si elles contiennent un ou plusieurs sites appropriés ;
- un site de fixation n'agit pas (à notre connaissance) seul, mais en coopération avec d'autres sites ayant des affinités avec d'autres facteurs ou avec une autre zone du même facteur. Aussi, la présence d'un site de fixation isolé, une *TATA-box* par exemple, ne représente peut-être pas toujours un début potentiel de transcription.

Ce caractère « exceptionnel » est donc à prendre avec une extrême prudence, et sans jamais oublier de le faire suivre de la question : « exceptionnel par rapport à quoi ? ». S'il est certain que les séquences nucléiques formant des sites de fixation ne sont pas réparties au hasard dans un génome, nous savons peu de choses des règles qui régissent cette répartition.

Coopérations entre sites

De nombreux exemples connus de sites de fixation, comme le consensus pour le facteur σ^{70} de *E.coli* ou les sites de régulation d'un gène chez *Drosophila* présentés précédemment, montrent des associations de sites à des distances contraintes. Il est très probable que des associations soient nécessaires pour initier une transcription. Nous avons noté qu'il existe peut-être un lien entre les contraintes de distances qui séparent ces sites et les structures et classes structurales des facteurs de transcription qui les reconnaissent. En l'absence d'information sur les facteurs mis en jeu, nous devons nous contenter de noter que le nombre et les distances qui séparent les sites de fixation peuvent être très variables. Certains organismes procaryotes présentent parfois des associations respectant des contraintes proches. Comme nous l'avons vu, les organismes eucaryotes font, eux, preuve d'une étonnante diversité et variabilité dans les sites mis en oeuvre, tant par leur nombre que par les distances qui les séparent et leur ordre.

Longueur et erreurs

On observe des longueurs très variables pour les sites de fixation : par exemple, la TATA-box est souvent représentée comme une succession de 6 nucléotides, parfois plus. Le site de fixation du facteur CRP chez *E.coli* est, lui, composé de 22 bases ([Combrugghe et al., 1984], [Berg and von Hippel, 1988]). On constate cependant dans certains cas qu'une ou plusieurs positions des sites de fixation, plus faibles, sont plus souvent sujettes à variations que d'autres. Ce caractère est encore une fois très variable selon les sites concernés : il a été observé que certains organismes ont des promoteurs mieux conservés que d'autres (voir par exemple [Mulder et al., 1997] au sujet des promoteurs de *E.coli*).

Conservation

Bien que les sites de fixation n'échappent pas au processus d'évolution, ils évoluent moins vite que les parties non-codantes du génome. Leur caractère fonctionnel les soumet en effet à une pression de sélection que le non-codant ne subit pas (voir [Tagle et al., 1988] et [Duret and Bucher, 1997]). On peut donc supposer que des organismes proches possèdent des sites de fixation communs, voire même des associations de sites communes et respectant des contraintes également communes. On dit que les sites de fixation ont un caractère conservé.

Caractère commun à des gènes co-régulés

L'expérience de la réponse SOS de *E.coli* est un exemple qui laisse supposer que certains gènes pourraient être transcrits simultanément parce que possédant des promoteurs et sites de régulation similaires, reconnus par un même facteur de transcription. En faisant l'hypothèse que des gènes co-régulés existent dans le génome d'un organisme, on suppose par conséquent qu'il sont précédés par des sites de fixation suffisamment similaires pour être reconnus par les mêmes facteurs. Une fois encore cependant, on sait peu de choses à ce jour concernant la co-régulation des gènes d'un organisme, et ce malgré l'apparition de techniques récentes (puces à ADN) permettant de relever les gènes qui y sont co-exprimés (car ils peuvent être exprimés simultanément pour des raisons indépendantes). On ne sait pas non plus l'ampleur et le type de différences qu'un facteur de transcription peut tolérer dans les sites de fixation qu'il reconnaît.

1.4 Discussion

L'exploration des propriétés propres aux sites de fixation laisse apparaître des caractéristiques extrêmement variées. Constatant cette diversité, il paraît bien ambitieux de vouloir décrire les spécificités des sites de fixation dans leur ensemble. Mais si l'usage du conditionnel semble être de rigueur pour décrire leurs propriétés générales, de nombreux travaux s'attachent à mettre en avant certaines propriétés de ces sites chez un organisme ou une famille d'organismes donnés ([Mulder et al., 1997], [Werner, 1999]), faisant état de certaines régularités qui nous donnent espoir de pouvoir détecter ces sites en développant des représentations et algorithmes adaptés.

Dans l'optique de réaliser cette détection par des algorithmes, le premier problème qui se pose à nous est descriptif. Plusieurs des propriétés énumérées décrivent des caractéristiques structurelles (longueurs, ordre et distances entre sites associés, nombre de sites associés) très variables. Les seules propriétés précises qui paraissent utilisables pour distinguer des sites de fixation sont leur caractère exceptionnel, la ressemblance entre certains sites de génomes ayant divergé, et la ressemblance entre sites reconnus par les mêmes facteurs de transcription. Les représentations informatiques qui décrivent des sites de fixation utilisent une ou plusieurs de ces propriétés pour réunir les sites qui les partagent, et renforcer ainsi le signal des sites face au bruit de la séquence. Le chapitre suivant présente les représentations les plus utilisées à ce jour.

Chapitre 2

De la biologie moléculaire à l'algorithmique du texte

Dans le chapitre précédent, nous avons décrit les propriétés biologiques connues et supposées des sites de fixation que nous nous sommes proposés de détecter. Le présent chapitre a pour but d'effectuer une transition entre la perception biologique de ces sites et leur représentation informatique. Cette étape de définition d'une représentation est très importante car elle détermine notre perception des sites de fixation, et, indirectement, les méthodes à utiliser pour les détecter.

Les représentations informatiques des sites de fixation utilisent la notion de ressemblance entre sites proches, ou bien le caractère exceptionnel d'un ensemble de sites, pour les réunir sous une même entité. Parmi ces deux propriétés, seule la notion de ressemblance peut être véritablement exprimée dans une représentation, indépendamment des séquences concernées. Le caractère exceptionnel est en effet fortement lié aux séquences manipulées, et fait donc intervenir des caractéristiques extérieures à la représentation. Dans ce chapitre, qui vise à décrire les représentations existantes indépendamment des algorithmes qui les manipulent, nous nous intéressons donc particulièrement à cette notion de ressemblance.

Les caractéristiques qui participent à établir une ressemblance entre sites de fixation, trop complexes et mal connues, ne peuvent être toutes prises en compte pour statuer. La difficulté à définir une bonne représentation utilisant la notion de ressemblance consiste à exprimer les particularités des sites de fixation qui permettent de les distinguer, tout en disposant d'une représentation suffisamment légère pour être aisément manipulée dans des algorithmes. Nous nous interrogeons par conséquent sur les critères pertinents à intégrer dans une représentation, sur la base des propriétés biologiques établies précédemment, du format des données génétiques que nous utilisons et de nos objectifs algorithmiques.

Dans une première section, nous présentons la matière première dont nous disposons pour traiter le problème qui nous intéresse. Les caractéristiques des représentations dépendent directement de la forme sous laquelle nous manipulons les séquences d'ADN.

Dans la section 2.2, nous définissons les conventions qui vont nous permettre de décrire les représentations existantes, et qui seront utilisées tout au long de ce mémoire.

Disposant des bases nécessaires, la question suivante est posée dans la section 2.3 : "comment regrouper des sites de fixation ?". Nous discutons d'abord des critères sur lesquels juger qu'une représentation regroupe des éléments de façon pertinente ou non. Les caractéristiques de la notion de ressemblance qui réunit des sites sous un même consensus sont ensuite discutées. Nous présentons les mesures couramment employées pour comparer deux sites de fixation. Enfin, nous passons en revue l'ensemble des représentations existantes afin de mettre en évidence leurs avantages et inconvénients respectifs.

2.1 L'ADN perçu comme un texte

La matière première dans laquelle nous cherchons à détecter des sites de fixation est constituée de macro-molécules d'ADN aux caractéristiques complexes. Nous distinguons :

- la séquence primaire, qui est la succession des symboles IUPAC (voir le tableau 1.1 page 20) représentant l'enchaînement des nucléotides d'un brin de l'ADN ;
- les propriétés physico-chimiques, les nucléotides ayant des caractéristiques qui concernent par exemple la force de certaines liaisons, leurs affinités entre eux ;
- les propriétés géométriques ou structurelles d'un fragment d'ADN.

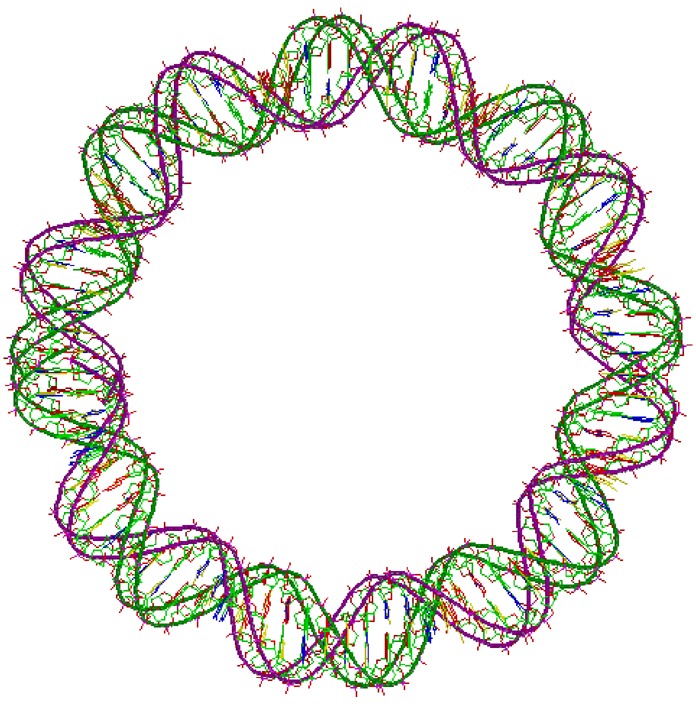

FIG. 2.1 – Acide désoxyribo-nucléique (ADN).
Modèle théorique. (J. L. Sussman, E. N. Trifonov).

La question qui vient le plus souvent à l'esprit des personnes à qui j'ai expliqué la teneur de mon travail et qui ne connaissaient pas la bio-informatique était : « sous quelle forme manipulez-vous ces séquences d'ADN ? Comment se présentent-elles à vous ? ». Mes interlocuteurs avaient souvent en tête l'image d'une séquence d'ADN modélisée par ordinateur, comme celle de la figure 2.1, telles qu'on les voit couramment dans les reportages scientifiques depuis la médiatisation de la génétique.

Aussi ont-ils souvent été surpris d'apprendre que ces séquences d'ADN sont réduites, dans le problème que nous traitons, à de simples fichiers de texte dans lesquels s'enchaînent les nucléotides en suivant un format précis (FASTA par exemple, voir la figure 2.2).

```
>ColE1
TAATGTTTGTGCTGGTTTTTGTGGCATCGGGCGAGAATAGCGCGTGGTGT
GAAAGACTGTTTTTTTGATCGTTTTCACAAAAATGGAAGTCCACAGTCTT
GACAG
> ARAC CONTIG 1 (BP 70073 TO 70178)
GAGTTGCGATAAAAAGCGTCAGGTAGGATCCGCTAATCTTATGGATAAAA
ATGCTATGGCATAGCAAAGTGTGACGCCGTGCAAATAATCAATGTGGACT
TTTCT
> TNAA CONTIG 317R (BP 270 TO 375)
TCAACCCTTTTTTTAAACATTAAAATTCTTACGTAATTTATAATCTTTAA
AAAAAGCATTTAATATTGCTCCCCGAACGATTGTGATTCGATTCACATTT
AAACA
> OMPA CONTIG 87R (BP 886 TO 941)
GTAGTTGAAAACTTACAAGTGTGAACTCCGTCAGGCATATGAAAAAAAAG
TCTTGTATAAGGTATGTTTAATCTTTTTTGTCAGCGACAATTTACAGAAG
AGAAT
```

Fig. 2.2 – Séquences d'ADN au format FASTA.
Les lignes commençant par ′ >′ sont réservées à l'annotation.

Cette simplification entraîne une perte d'information puisque, ainsi représenté sous forme de texte, un fragment d'ADN perd toutes ses caractéristiques de conformation spatiale et ses propriétés physico-chimiques. Elle est pourtant imposée par le fait que les données rendues disponibles en masse depuis l'intensification du séquençage d'ADN ne concernent que sa structure primaire, c'est-à-dire sa séquence. Il existe certes des séquences d'ADN dont on a décrit de plus larges caractéristiques, mais en très petit nombre (comparé au produit des séquençages actuels). En particulier, les propriétés de conformation spatiale des séquences d'ADN restent assez mal connues à ce jour.

Cette perte d'information nous sert cependant, puisqu'elle réduit considérablement la complexité du problème à traiter, et le rend abordable par l'algorithmique du texte. Les choix des caractéristiques à donner aux sites de fixation se voient en effet fortement réduits.

Ces sites, d'objets mystérieux pour les algorithmiciens du texte, prennent alors une forme qui leur est bien connue : les mots.

Reste une question de taille : cette vision réductrice de l'ADN contient-elle tout de même les informations nécessaires à la mise en évidence des sites de fixation ? Au vu des connaissances actuelles dans ce domaine, et comme nous le verrons dans les applications du dernier chapitre, la séquence primaire de l'ADN contient suffisamment d'informations pour mettre en évidence certains sites de fixation, mais probablement pas tous. Malgré l'aspect restrictif des séquences d'ADN telles que nous les percevons, il pourrait exister une corrélation plus ou moins forte entre un nucléotide et une ou plusieurs propriétés physico-chimiques ou géométriques. De ce fait, il est fort possible que, en nous restreignant au niveau de la séquence primaire, nous prenions implicitement en compte de telles propriétés.

2.2 Conventions

La bio-informatique, croisement des domaines de l'informatique et de la biologie, est un domaine de recherche relativement jeune. Si de nombreux algorithmes utilisés en bio-informatique empruntent leurs bases à un domaine un peu plus ancien, l'algorithmique du texte, ce transfert de connaissances est assez récent. L'intersection de ces deux domaines impose de faire des choix pour désigner des objets appartenant aux deux mondes. Nous nous attachons donc ici à définir clairement les termes employés afin de ne pas introduire d'ambiguïtés pour les lecteurs provenant de ces deux domaines.

2.2.1 Vocabulaire de base

Nous venons de voir que les séquences d'ADN sont, dans notre cas, perçues comme de simples *textes* (ou *chaînes de caractères*), selon la nomenclature utilisée en algorithmique du texte. Dans ce domaine, le terme de séquence utilisé en biologie n'a pas de signification précise. En accord avec la terminologie utilisée en génétique, nous désignerons par *séquences* les données sur lesquelles nous travaillons.

Les séquences utilisées pour inférer des sites de fixation sont des séquences d'ADN, dont les composants sont des bases nucléiques. Les problèmes d'inférence dont nous traitons dans ce mémoire peuvent s'appliquer à des séquences protéiques composées d'acides aminés ou à n'importe quel autre type de séquences. Quels qu'ils soient, nous désignerons par *symboles* [1] les éléments des séquences utilisées. Ces symboles sont définis sur un *alphabet* fini, par exemple ceux des nucléotides ou des acides aminés.

Des suites de symboles successifs forment des *mots*. Nous nommerons *occurrences* (appelées *facteurs* en algorithmique du texte) d'un mot les différentes apparitions de ce mot dans une séquence.

[1] le terme de symbole est emprunté à l'algorithmique du texte, l'entité équivalente en biologie serait le *monomère*.

Par exemple, le mot **GAGC** a deux occurrences dans la séquence :

ATC**GAGC**ATAGGACGATCTAGCA**GAGC**ATCATCGACA

Si ces termes permettent de désigner les objets que nous manipulons, il leur faut une définition algorithmique précise afin de pouvoir les utiliser de façon rigoureuse.

2.2.2 Définitions algorithmiques

Tout au long de ce mémoire, nous considérerons que nous travaillons sur un alphabet de symboles nommé Σ (dans le cas d'une séquence d'ADN, nous aurons par exemple $\Sigma = \{A, C, G, T\}$).

On note usuellement Σ^*, l'ensemble de tous les mots finis composés de symboles appartenant à Σ. Le mot vide est noté ε. De la même manière, on note Σ^+ l'ensemble Σ^* privé du mot vide (en notation ensembliste $\Sigma^+ = \Sigma^* \backslash \{\varepsilon\}$). Enfin, Σ^k désigne l'ensemble des mots de longueur k.

Étant donnée une séquence $s \in \Sigma^+$, nous posons les conventions suivantes :
- $|s|$ est le nombre de symboles de la séquence s ;
- $s[i]$ désigne le $i^{\text{ème}}$ symbole de la séquence s, pour $1 \leq i \leq |s|$;
- $s[i...j]$ désigne l'occurrence comprise entre les positions i et j (incluses) de la séquence s, pour $1 \leq i \leq j \leq |s|$;
- le mot $w \in \Sigma^+$ a une occurrence dans s si il existe deux mots $x, y \in \Sigma^*$ tels que $s = xwy$ (où xwy désigne la concaténation de ces trois mots) ;
- on dit que « le mot w a une occurrence dans la séquence s à la position i » si on a

$$\forall j \in [1, |w|], s[i + j - 1] = w[j].$$

2.3 Comment regrouper des sites de fixation ?

La représentation des séquences que nous manipulons a été définie. Il nous faut maintenant faire de même avec les sites de fixation que nous souhaitons détecter. La simplification appliquée aux séquences d'ADN dont nous disposons s'applique évidemment aussi aux sites de fixation qu'elles peuvent contenir. Ces derniers sont donc dorénavant perçus comme des mots.

Notre objectif est de développer des concepts et outils suffisamment efficaces pour mettre en évidence des groupes d'objets issus d'une même famille fonctionnelle et qu'on suppose donc partager des propriétés communes. Dans le problème biologique qui nous intéresse, ces objets sont des promoteurs ou des sites de régulation, que nous réunirons sous le terme de « sites ». La réunion de ces sites sous une même entité est un premier pas essentiel vers la compréhension des mécanismes de régulation. C'est en effet en cherchant à réunir ces sites que nous pourrons peut-être mettre en évidence les caractéristiques qu'ils partagent. Si ces regroupements de sites sous un objet commun sont susceptibles de nous faire perdre des informations sur leurs caractéristiques individuelles, ils peuvent nous permettre de dégager le « patron » qui expliquerait des fonctionnalités communes.

Ce regroupement se justifie aussi d'un point de vue algorithmique puisqu'il est beaucoup plus aisé de manipuler une entité englobant plusieurs sites que chacun de ces sites séparément.

Dans un premier temps, nous définissons les propriétés que nous estimons indispensables à une bonne représentation. Puis, nous étudions la notion de ressemblance qui permet à certaines représentations de réunir des sites « proches ». Enfin, nous présentons les représentations existantes, qui utilisent ou non la ressemblance pour regrouper des sites.

2.3.1 Propriétés d'une « bonne » représentation

Imaginons que nous disposons de séquences dans lesquelles un certain nombre d'occurrences d'un site connu ont été trouvées. Ces occurrences présentent des propriétés telles que l'on souhaite les réunir dans un même ensemble associé à ce site de fixation. La question que nous désirons éclaircir est « quelles propriétés donner à cet ensemble afin qu'il décrive au mieux ce site et ses occurrences ? ».

Afin d'étayer notre propos, essayons-nous au jeu de la figure 2.3 : il consiste à trouver celle des trois figures a, b ou c qui permet d'attraper le maximum de points rouges et le minimum de points verts en la positionnant à l'intérieur du grand rectangle. Après quelques essais, on s'aperçoit rapidement des forces et des faiblesses de chacune des trois figures :

- le rectangle a est très grand et permet d'attraper beaucoup de points rouges, mais nous oblige à attraper également un certain nombre de points verts ;
- le cercle b est de taille plus réduite et sa forme nous permet d'attraper moins de points rouges que le rectangle mais un seul point vert ;
- le triangle c est le plus petit. On ne peut attraper que peu de points rouges avec, mais sans attraper de points verts.

Parallèlement à notre problème, notons que le fait de découvrir que c'est, par exemple, le cercle qui est le mieux adapté pour recueillir des points rouges nous apprend implicitement que les points rouges ont peut-être pour propriété d'être réunis en cercle. De même, le regroupement de sites sous une même entité peut nous apprendre certaines caractéristiques de ces sites.

Ce jeu amène naturellement une question qui concerne la façon dont le problème est posé : qu'est-ce qui vaut le mieux ?

- attraper le maximum de points rouges ? On privilégie alors la *sensibilité*, qui est l'aptitude à détecter le critère choisi ;
- éviter le maximum de points verts ? On privilégie alors la *spécificité*, qui est l'aptitude à ne détecter que le critère choisi ;
- un compromis entre les deux ? Puisque sensibilité et spécificité maximales semblent incompatibles dans ce cas et avec les figures dont nous disposons.

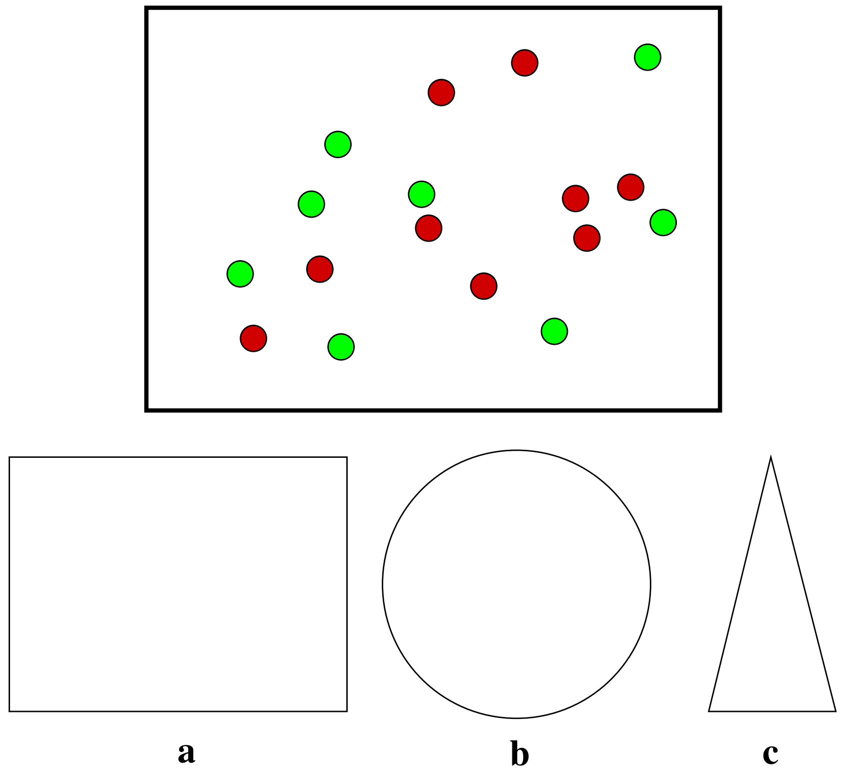

FIG. 2.3 – Un petit jeu.
Avec laquelle de ces trois figures peut-on attraper le maximum de points rouges (foncés) et le minimum de points verts (clairs) ? Cette question, qui nécessite d'être précisée, introduit le compromis à faire entre sensibilité et spécificité.

Les mêmes questions s'appliquent à la représentation de sites que nous cherchons à définir : dans quelle mesure doit-elle privilégier sensibilité ou spécificité (qui s'avèrent en fait incompatibles en général) ?

Enfin, pour résoudre ce jeu de manière automatique, nous décidons d'élaborer une tactique visant à produire la meilleure solution. Avec un cercle de rayon variable, on pourra par exemple appliquer une méthode simple qui consiste à se placer au barycentre d'un ensemble de points rouges et à faire varier le rayon en essayant d'optimiser le résultat. Mais si, par souci d'épouser plus fidèlement un groupe de points rouges pour augmenter sensibilité et spécificité, on use d'une forme plus complexe, un polygone irrégulier par exemple, la tactique à adopter pour visiter toutes les solutions ne sera-t-elle pas également plus complexe ?

Par ce dernier constat, nous voulons mettre en avant le lien fort qui existe entre la complexité de la représentation choisie et la complexité de l'algorithme qui va résoudre un problème de détection en utilisant cette représentation. Ce lien existe également entre la représentation que nous devons choisir pour décrire des sites et les algorithmes qui vont les utiliser. Cette représentation doit donc être suffisamment simple pour que l'algorithme employé et qui va « visiter » l'ensemble des solutions ne soit pas rendu trop coûteux.

Une représentation des sites que nous souhaitons décrire doit donc être jugée sur :
- sa sensibilité ;
- sa spécificité ;
- sa complexité.

Le jeu des points rouges et verts nous a permis de mettre en évidence certaines propriétés importantes pour la représentation que nous sommes amenés à utiliser. Il est malheureusement assez loin du problème qui nous concerne, puisque les vraies et fausses occurrences de sites ne nous apparaissent pas aussi clairement que les points rouges et verts (les moyens dont nous disposons pour tenter de les distinguer seront discutés dans la section 3.3 du prochain chapitre). Les occurrences des sites que nous cherchons à mettre en commun ne sont pas non plus nécessairement identiques entre elles.

Deux des propriétés supposées des sites de fixation (leur conservation et leur caractère commun à des gènes co-régulés) impliquent une notion de ressemblance, qui est exploitée par les représentations, de façon plus ou moins forte, pour réunir des sites. Nous traitons donc de cette notion, dont on trouvera une exploration plus générale dans [Sagot, 1996].

2.3.2 De la ressemblance de deux occurrences

Dire que deux objets se ressemblent implique intrinsèquement l'existence d'une méthode de comparaison, connue ou non. Comme deux personnes aux sensibilités différentes, l'une visuelle et l'autre auditive, établiront des associations différentes à partir d'une même scène, une méthode de comparaison n'impliquera pas nécessairement les mêmes associations qu'une autre. On ne peut pas dire pour autant qu'une méthode soit meilleure qu'une autre dans l'absolu. Certaines méthodes sont cependant plus pertinentes que d'autres dans un contexte particulier.

Illustrons à nouveau le contexte de notre problème, en considérant cette fois une foule dans laquelle nous souhaitons réunir les personnes qui se ressemblent. En premier lieu, nous devons choisir sur quelles données nous comparons ces personnes : se contente-t-on d'observer chaque personne ou de l'écouter, ou bien s'entretient-on plus longuement avec elle afin de connaître ses aptitudes, son parcours ?

À partir de ces données, quels critères utilisons-nous ensuite pour effectuer cette comparaison ? Sur un visage, doit-on tenir compte de la couleur des yeux, de la forme du visage ? Est-ce que le fait que deux personnes aient des métiers proches est important ou pas ?

Enfin, quelle souplesse s'autorise-t-on sur chacun des critères choisis ?

Les réponses à ces questions sont évidemment dépendantes du contexte et de la requête formulée. En particulier, le temps que nous souhaitons consacrer à cette comparaison et leur objectif sont des critères primordiaux pour y répondre. Dans le cas où le temps est compté, on doit réduire autant que possible le nombre et la complexité des critères de comparaison. Si la requête est complexe, il faut au contraire les augmenter.

Dans notre problème, la question des données sur lesquelles nous effectuons les comparaisons est déjà résolue : de par le format sous lequel nous disposons des séquences d'ADN, nous percevons les sites de fixation comme des mots. Encore une fois, il est rare de disposer d'autres données sur des séquences d'ADN qui pourraient être utilisées pour comparer deux sites.

Pour comparer des mots, on utilise couramment des distances d'édition, qui permettent de quantifier le nombre d'opérations permettant de transformer un mot en un autre. On pourrait toutefois utiliser d'autres critères de comparaison sur les mots, plus simples ou plus complexes. Les distances d'édition ont pour avantage d'offrir un critère de comparaison qui peut être pertinent, et qui est généralement peu coûteux.

Comme toute méthode de comparaison, celle-ci générera ses associations propres, qui ne correspondent pas nécessairement à celles qui lient des sites de fixation ressemblants, reconnus par le même facteur de transcription ou issus d'organismes ayant un proche ancêtre commun. C'est pourquoi nous devons garder à l'esprit qu'un biais est ici introduit à nouveau. Notons dès lors que cette méthode de comparaison, simpliste, aura probablement tendance à créer des regroupements trop larges, manquant de spécificité. Une représentation utilisant cette méthode de comparaison dispose cependant d'assez de souplesse (par le biais de la distance maximum qu'on autorise entre deux mots) pour ajuster la taille de l'ensemble d'occurrences lui appartenant.

Au terme de nos travaux et d'applications à des données biologiques, nous discuterons plus avant de la validité de l'hypothèse qui consiste à dire qu'une distance d'édition entre mots est suffisante pour détecter des sites de fixation ressemblants. Nous traiterons également des données supplémentaires qui pourraient raisonnablement être utilisées dans des comparaisons (aspects physico-chimiques, conformation spatiale du fragment d'ADN ou distances évolutives d'un arbre phylogénétique [2]), dans le but d'affiner notre méthode de comparaison entre mots. Ces améliorations devront évidemment se faire sans perdre de vue l'incidence cruciale de la méthode de comparaison sur la complexité de l'algorithme qui l'utilise.

Présentons maintenant les distances d'édition les plus couramment utilisées.

2.3.2.1 Distance de Hamming

La ressemblance entre deux mots peut être mesurée par la quantité, le coût et le type des opérations à effectuer pour transformer l'un en l'autre. On parle alors de *distance* [3]

[2] un arbre phylogénétique est une représentation des liens évolutifs entre différentes espèces fossiles et actuelles, compte tenu de leurs places respectives dans la chronologie.

[3] une distance d sur un ensemble E respecte les propriétés suivantes pour tout $a, b, c \in E$:
- $d(a,b) \geq 0$
- $d(a,b) = d(b,a)$
- $d(a,b) = 0$ si et seulement si $a = b$

d'édition entre deux mots.

Parmi les altérations pouvant affecter un mot, la plus fréquemment rencontrée dans des séquences génétiques est la *substitution* d'un symbole par un autre. La *distance de Hamming* mesure le nombre minimal de substitutions entre deux mots de même longueur [4] (voir figure 2.4).

Notation 1 (Distance de Hamming)
Soient deux mots $u, v \in \Sigma^*$ *tels que* $|u| = |v|$, *on note* $\mathcal{D}_H(u, v)$ *la distance de Hamming entre ces deux mots.*

Le calcul de la distance de Hamming entre deux mots de longueur n se fait en temps linéaire $\mathcal{O}(n)$.

```
A C T T G C A G T T C A G G G

C C T T C G C G T T C A C G A
```

FIG. 2.4 – Distance de Hamming entre deux mots.
Les substitutions effectuées entre chacun des deux mots sont encadrées. La distance qui les sépare vaut 6.

2.3.2.2 Distance de Levenshtein

La *distance de Levenhstein* ([Levenshtein, 1966]) ajoute deux opérations sur les mots aux seules substitutions autorisées par la distance de Hamming :
- l'*insertion*, qui consiste à ajouter un symbole à une position donnée d'un mot ;
- la *délétion*, ou *suppression*, qui supprime un symbole de la même façon.

Il est d'usage d'employer les mots anglais *indel* pour désigner ces deux opérations et *gap* pour désigner une suite d'*indels*, et le mot *erreur* [5] une des trois opérations d'édition (substitution, insertion ou délétion). La distance de Levenshtein mesure le nombre minimal d'erreurs nécessaires pour transformer un mot en un autre. Notons que cette distance s'applique cette fois entre mots de longueurs quelconques.

En utilisant la distance de Levenshtein (et contrairement à la distance de Hamming), on peut parfois obtenir la même distance entre deux mots de plusieurs façons (*i.e.* en utilisant différentes successions d'opérations, voir l'exemple figure 2.5).

– $d(a, b) + d(b, c) \geq d(a, c)$

[4] la distance de Hamming entre deux mots de longueurs différentes n'a pas de sens.

[5] le terme de *mutation* est employé en biologie pour désigner une erreur.

FIG. 2.5 – Distance de Levenshtein entre deux mots.
Entre les mots $AGTCAAT$ et $AGATAATA$, la distance de Levenshtein vaut 3. On peut obtenir ce résultat de deux façons :
(a) une délétion du C dans le premier mot, et deux insertions d'un A ;
(b) deux substitutions et une insertion dans le premier mot.

Notation 2 (Distance de Levenshtein)
Soient deux mots $u, v \in \Sigma^*$, on note $\mathcal{D}_L(u, v)$ la distance de Levenshtein entre ces deux mots.

Le calcul de la distance de Levenshtein se fait par programmation dynamique (voir [Needleman and Wunsch, 1970]), avec un coût en temps et en espace en $\mathcal{O}(nm)$ où n et m sont les longueurs des deux mots comparés. Plusieurs algorithmes ([Hunt and Szymanski, 1977], [Myers, 1986] et [Ukkonen, 1985]) offrent une meilleure complexité en moyenne, mais la complexité dans le pire des cas de ce type de comparaison reste quadratique, ce qui en fait une distance assez coûteuse à utiliser.

2.3.2.3 Distance pondérée et matrices de substitution

Une variante de la distance de Levenshtein consiste à pondérer les trois opérations d'édition, ceci afin de leur donner des priorités. Ce type de pondération a un sens biologique, puisque les fréquences d'apparition de ces trois opérations d'édition ne sont en général pas les mêmes (les substitutions sont par exemple plus fréquentes que les *indels*).

Si nous reprenons l'exemple de la figure 2.5 en donnant un coût de 1 pour la substitution et de 2 pour la délétion et l'insertion, le coût des opérations faites en (a) est maintenant de 6, alors qu'il est de 4 en (b). Les opérations faites en (b) représentent alors le coût minimal pour cet exemple.

D'autres variantes donnent un coût plus élevé à la première insertion ou délétion (coût d'ouverture de *gap*) d'une série d'*indels*. Le coût d'insertion est alors une fonction affine dépendant de la longueur du *gap*.

On peut également donner un coût à chaque opération en fonction des symboles manipulés. Le coût d'une insertion/délétion, dépendant alors du symbole inséré/délété, est donné par une fonction $f : \Sigma \to \mathbb{N}$ (définie par un vecteur de dimension $|\Sigma|$) associant un coût à chaque symbole de l'alphabet. De même, le coût d'une substitution est donné par une fonction $f : \Sigma \times \Sigma \rightarrow \mathbb{N}$ (définie par une matrice symétrique, de dimension $(|\Sigma|, |\Sigma|)$), qui donne un coût à chaque paire de symboles de l'alphabet.

De telles matrices de substitution sont surtout utilisées quand on manipule des séquences protéiques (matrices PAM [Dayhoff, 1978] et Blosum [Henikoff and Henikoff, 1992]) car les acides aminés possèdent des propriétés plus complexes que les nucléotides, qui créent des relations particulières entre eux (voir [Sagot, 1996]). Elles indiquent, pour des protéines étant à une certaine distance évolutive, la fréquence à laquelle deux acides aminés différents peuvent être observés à la même position.

2.3.3 Les représentations existantes

Nous passons maintenant en revue les différentes représentations existantes qui permettent de réunir plusieurs sites respectant une propriété.

Deux types de représentations ont été principalement utilisées à ce jour pour décrire ces sites spécifiques. La première regroupe motifs et matrices, la seconde modèles de Markov et réseaux neuronaux. Les motifs sont les seules représentations présentées qui utilisent une notion de ressemblance stricte, déterministe, telle que nous l'avons définie à l'aide de distances, pour réunir des sites.

2.3.3.1 Motifs et matrices

Motifs consensus simples
La façon la plus naturelle et la plus ancienne de regrouper plusieurs mots proches consiste à employer la notion de *motif consensus* [6]. Un motif est un ensemble qui regroupe certaines occurrences d'une séquence respectant une contrainte de distance précise.

Définition 1 (Motifs et occurrences)
Étant donnés un motif $m \in \Sigma^*$, une séquence $s \in \Sigma^*$, une distance $\mathcal{D}$ et un seuil maximal de distance e, tous les mots $x \in \Sigma^*$ de la séquence s tels que $\mathcal{D}(m, x) \leq e$ sont appelés des occurrences du motif *m dans la séquence s. L'occurrence d'un motif est dite* exacte *quand la distance qui la sépare de ce motif est nulle, et* approchée *sinon.*

Notation 3 (Motif consensus)
On notera m^e un motif consensus m autorisant une distance maximale e.

Un motif est un objet extérieur à une séquence : il peut avoir des occurrences approchées dans cette séquence, sans pour autant y apparaître lui-même. Par exemple, en utilisant une distance de Hamming, dans la séquence

$$A\underline{CAAT}A\overline{GCG}TTA\overline{ACC}A\overline{GTC}T\overline{GCC}$$

le motif GCC^1 possède une occurrence exacte et trois occurrences approchées (surlignées). Le motif $AAAT^1$ a seulement une occurrence approchée (soulignée) ; il n'apparaît pas exactement dans la séquence.

[6] *pattern* en anglais. Les termes *motif* et *consensus* seuls sont souvent utilisés pour désigner des motifs consensus.

La notion de motif amène naturellement à celle de *voisinage d'un motif* (voir figure 2.6) :

Définition 2 (e-voisinage)[Myers, 1994]
Étant donnée une distance $\mathcal{D}$, le e-voisinage $V_e(m)$ *d'un motif* m *est l'ensemble de tous les mots qui sont au plus à une distance* e *de ce motif. On a donc :*

$$V_e(m) = \{x \in \Sigma^* / \mathcal{D}(m, x) \leq e\}$$

$\mathcal{V}(e, k)$ *est le nombre de mots distincts du e-voisinage d'un motif de longueur* k*, c'est-à-dire le cardinal de son e-voisinage.*

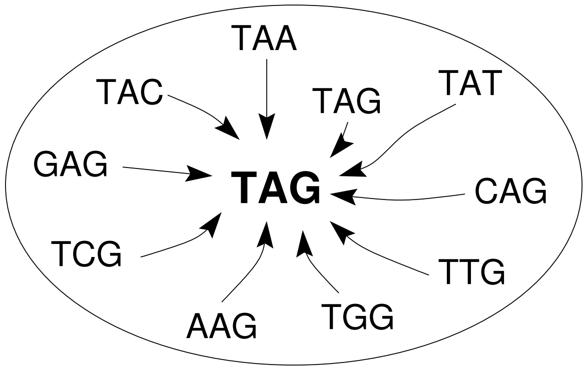

FIG. 2.6 – e-voisinage d'un motif consensus.
Le motif TAG[1] et les 10 mots ($\mathcal{V}(1,3) = 10$) qui se trouvent dans son 1-voisinage (noté $V_1(TAG)$) si on utilise la distance de Hamming.

Lemme 1
Si la distance utilisée est celle de Hamming on a :

$$\mathcal{V}(e, k) = \sum_{j=0}^{e} \binom{k}{j} (|\Sigma| - 1)^j.$$

Preuve : Ici, j est le nombre de substitutions possibles, il varie donc de 0 à e. Pour une substitution à une position donnée du mot m, il existe $|\Sigma| - 1$ symboles pouvant se substituer au symbole original. Si on autorise exactement j substitutions à des positions quelconques dans le mot, il existe $\binom{k}{j} (|\Sigma| - 1)^j$ mots x tels que $d(x, m) = j$. Pour e substitutions au plus, le e-voisinage de m contient par conséquent $\sum_{j=0}^{e} \binom{k}{j} (|\Sigma| - 1)^j$ mots distincts. □

Un motif consensus est une représentation de type déterministe : étant donné un motif consensus m et une distance maximale e, on peut dire de tout mot sur le même alphabet qu'il respecte ou ne respecte pas le consensus m.

Motifs consensus étendus
On peut aussi définir un motif consensus sur un alphabet dégénéré [7]. Typiquement, pour des séquences d'ADN, on utilise l'alphabet normé IUPAC (tableaux 1.1 et 1.2, page 20), qui prend en compte les relations physico-chimiques qui lient certains nucléotides.

Sur l'alphabet des nucléotides A, C, G, T, on utilise par exemple les symboles R (purines) et Y (pyrimidines) pour désigner respectivement « A ou G » et « C ou T ». Les motifs ainsi construits sont en fait des expressions régulières simples, uniquement composées à partir d'unions et concaténations (*i.e.* sans fermeture de Kleene) : $GATRTY$ s'écrit $GAT[AG]T[CT]$. Myers les nomme *network expressions* ([Myers and Miller, 1989]).

Ces expressions peuvent également utiliser des *jokers* (notés N sur l'alphabet des nucléotides et X sur celui des acides aminés) et des *spacers*, qui sont respectivement des symboles et suites de symboles spéciaux représentant n'importe quel symbole de l'alphabet. Les *spacers* sont particulièrement adaptés à la description de sites associés et séparés par une distance contrainte (comme ceux décrits dans la figure 1.3 page 27).

On peut décider d'utiliser un alphabet dégénéré simple (comme A, C, G, T, R, Y) ou plus complet pour décrire un site. Cependant, il y a ici un compromis à faire dans le choix du langage utilisé. En effet, le choix d'un langage très dégénéré permet de représenter des motifs moins contraints, créant des associations plus grossières et regroupant donc plus d'occurrences, et de prendre en compte (de façon limitée) certaines propriétés physico-chimiques que l'on ignore sinon. En contrepartie, le nombre d'éléments d'un tel langage est plus élevé que celui d'un langage simple, il devient donc plus coûteux d'élaborer un algorithme exact explorant l'intégralité de l'espace des solutions d'un problème de détection utilisant cet alphabet.

Brazma *et al.* donnent une classification des langages permettant de définir des motifs consensus en fonction de leur complexité ([Brazma et al., 1998a]).

Les motifs consensus, étendus ou pas, bien que couramment utilisés, sont considérés comme inadéquats pour représenter la spécificité d'un site de fixation ([Bucher et al., 1996], [Hertz and Stormo, 1996]). Biologiquement, le principal reproche qui leur est fait est le suivant : pour chacune de ses positions, un motif consensus ne fournit que le symbole le plus représenté parmi l'ensemble d'occurrences qu'il décrit. On peut percevoir un motif consensus comme une boule dont le motif serait le centre et dont le rayon serait déterminé par le nombre d'erreurs autorisé. Les occurrences de ce motif se trouvent à l'intérieur de la boule. Le passage d'un ensemble d'occurrences au motif consensus auquel elles appartiennent constitue une perte d'information puisque le consensus ne permet pas de savoir comment sont réparties les occurrences dans la boule.

Ils autorisent l'utilisation de *gaps*, mais ne permettent pas d'exprimer des corrélations entre des positions d'un site.

Par rapport aux critères que nous avons énoncés comme nécessaires à une bonne re-

[7]un alphabet dégénéré est un alphabet des motifs dans lequel certains symboles peuvent représenter plusieurs symboles de l'alphabet des séquences (un sous-ensemble des sous-ensembles de l'alphabet).

présentation, ils disposent d'une certaine flexibilité permettant d'obtenir une sensibilité correcte, mais permettent difficilement d'obtenir une bonne spécificité.

L'utilisation de motifs consensus autorise cependant le développement d'algorithmes très efficaces de par la simplicité de représentation et de comparaison qu'ils permettent. C'est pourquoi, ne perdant pas de vue leurs limitations et essayant d'y pallier, nous avons choisi d'utiliser des objets proches des motifs consensus dans nos travaux.

Matrices pondérées

La forme la plus couramment employée pour représenter la spécificité d'un site de fixation est la matrice pondérée. La notion de ressemblance employée est ici beaucoup moins rigide que celle utilisée dans les motifs. Pour décrire les matrices pondérées, il nous faut d'abord expliquer ce que sont un *profil* et une matrice *log-likelihood*.

Une matrice profil P est une matrice de dimension $\Sigma \times k$, où k est la longueur des occurrences que l'on souhaite représenter. Elle contient dans chacune de ses cases $P[\alpha, i]$ la fréquence $F_{\alpha,i}$ d'apparition du symbole $\alpha \in \Sigma$ à la $i^{\text{ème}}$ position dans l'ensemble des sites modélisés (figure 2.7).

CTAAGG
CTAAGT
CTGAGT
CTAAGG
TTGAAG

	1	2	3	4	5	6
A	0	0	0.6	1	0.2	0
C	0.8	0	0	0	0	0
G	0	0	0.4	0	0.8	0.6
T	0.2	1	0	0	0	0.4

FIG. 2.7 – Un ensemble d'occurrences et son profil.
Le motif consensus le plus proche de ce profil serait $CT[AG]AG[GT]$ (où $CTRAGK$).

Dans une matrice *log-likelihood*, on remplace la fréquence $F_{\alpha,i}$ de chaque case par $\log \frac{F_{\alpha,i}}{G_\alpha}$, où G_α est la fréquence globale d'apparition du symbole α dans la séquence d'où proviennent les occurrences à représenter. Ce passage au *log* permet de transformer le produit des fréquences à effectuer lors d'un alignement en somme, plus aisément calculable et d'un ordre de grandeur plus praticable. Une matrice *log-likelihood* n'est plus une entité indépendante permettant de réunir des occurrences : les coefficients qu'elle contient dépendent de la composition des séquences manipulées.

Quand on dispose d'une telle matrice, on évalue si une occurrence répond ou non à ses caractéristiques en réalisant un alignement entre la matrice et cette occurrence. Quand on aligne une matrice *log-likelihood* M de dimension $\Sigma \times k$ avec une position donnée d'une séquence, on obtient un score pour l'occurrence de taille k se trouvant à cette position en sommant les valeurs rencontrées à chaque case de la matrice P pour cet alignement (voir figure 2.8). Ce score est connu sous le nom de *log-likelihood ratio*. Plus il est élevé, plus la corrélation entre le site représenté par la matrice et l'occurrence proposée est grande.

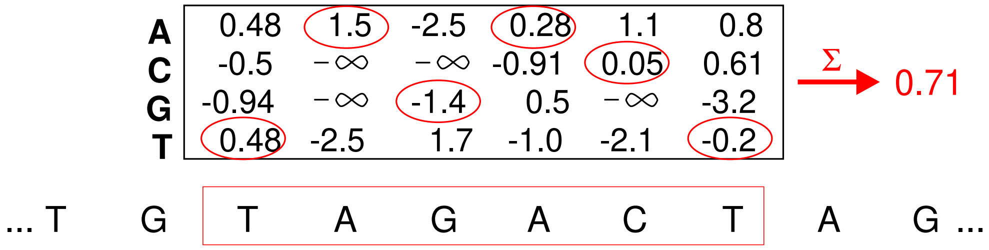

Fig. 2.8 – Alignement d'une matrice *log-likelihood* et d'une séquence.
Le *log-likelihood ratio* obtenu pour l'occurrence TAGACT est de 0.71.

Mulligan *et al.* ont montré qu'il existe une forte corrélation entre les valeurs des scores obtenus par l'alignement d'une matrice *log-likelihood* et le niveau d'activité des sites de fixation décrits par cette matrice ([Mulligan et al., 1984]). Berg *et al.* ont fait une étude théorique qui confirme ces résultats ([Berg and von Hippel, 1987], [Berg and von Hippel, 1988]).

Une *matrice pondérée* est une matrice M de dimension $\Sigma \times k$ permettant d'assigner à un mot $m \in \Sigma^k$ un score obtenu par le calcul suivant : $\sum_{i=1}^{k} M[m[i], i]$. Les matrices *log-likelihood* sont des matrices pondérées. La démarche de construction d'une matrice pondérée nécessite de disposer d'un jeu d'occurrences du site que l'on souhaite représenter.

Les matrices pondérées autorisent une grande finesse de description de chacune des positions du site représenté : contrairement aux motifs consensus, les positions d'une matrice pondérée peuvent donner préférence à plusieurs symboles en leur affectant des poids. Une matrice pondérée a ainsi pour avantage d'« épouser » un ensemble d'occurrences plus finement que les motifs consensus, moins souples. À ce titre, les matrices pondérées sont des représentations pouvant apporter plus de spécificité que les motifs consensus. Il a cependant été montré que leur capacité de description de sites de fixation reste assez médiocre ([Hertz and Stormo, 1996], [Duret and Bucher, 1997]). La représentation matricielle n'apporte pas que des avantages puisqu'elle impose une certaine rigidité de la structure des sites qu'elle décrit ; elle ne permet pas en général de gérer des *gaps* (Hertz et Stormo ont cependant proposé une méthode d'inférence qui utilise une représentation matricielle et prend en compte les *gaps*, et ce sans avoir à déterminer leurs coûts au préalable [Hertz and Stormo, 1994]). Comme les motifs, les matrices ne peuvent exprimer, sous leur forme originale, de corrélations entre différentes positions (par exemple, « les positions 2 et 5 ne doivent contenir au total qu'un seul A »).

Enfin, et c'est leur inconvénient majeur, elle sont, du fait de leur souplesse, d'une complexité plus élevée que les motifs consensus : dans le cadre d'une inférence, l'espace des solutions qu'elles décrivent (l'ensemble de tous les alignements possibles) est généralement beaucoup plus vaste que celui décrit par les motifs. Leur souplesse se transforme alors en inconvénient, et impose de faire usage d'autres propriétés, comme le caractère exceptionnel des sites, pour réunir des occurrences. Nous traiterons des scores permettant d'estimer la pertinence d'une matrice dans le chapitre suivant.

2.3.3.2 Modèles de Markov et réseaux neuronaux

Les chaînes de Markov sont des processus stochastiques qui permettent d'exprimer la dépendance d'un évènement donné avec les k évènements précédents. L'ordre d'une chaîne de Markov correspond à la valeur choisie pour k. Les chaînes de Markov ont été largement utilisées dans le cadre de la prédiction de gènes, qui consiste à distinguer les régions codantes des régions non-codantes de l'ADN en analysant la composition locale en mots de longueur fixée le long d'une séquence. Du fait de leurs évolutions différentes, il existe souvent un biais de composition entre régions codantes et non-codantes qui permet cette détection.

Pour la représentation de sites particuliers, les modèles de Markov cachés (HMM en abrégé anglais, voir l'exemple de la figure 2.9), ont parfois été employés. On en trouvera une description détaillée ainsi que des problèmes associés dans [Bréhelin and Gascuel, 2000].

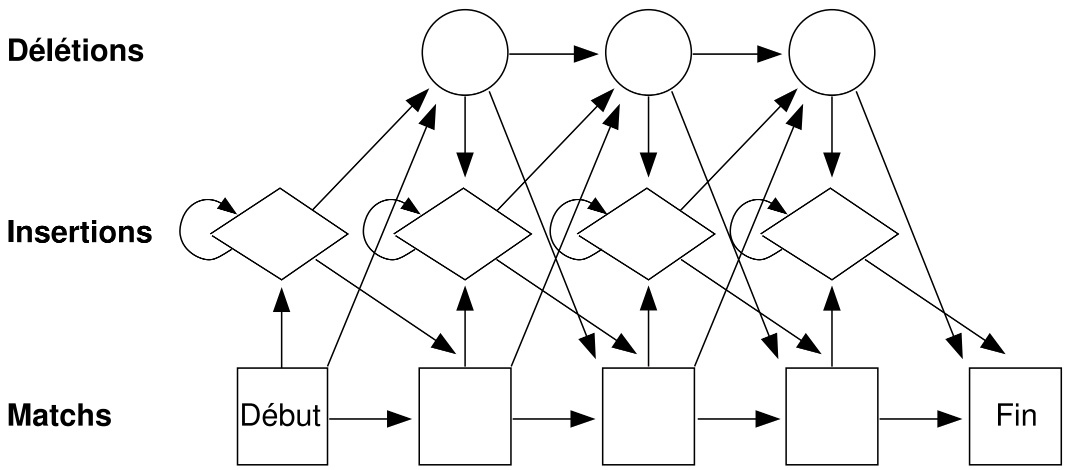

FIG. 2.9 – Un exemple de HMM.
À chaque transition est associée une probabilité. Les états *match* et insertion émettent des caractères avec une certaine probabilité, en fonction des k caractères émis précédemment (k est l'ordre du modèle de Markov).

Ce sont des automates stochastiques non déterministes (une même séquence de symboles peut être générée de plusieurs manières, et ce sans que l'on sache quel chemin a été emprunté). Les symboles produits dans chaque état et les transitions qui en sortent peuvent dépendre de chaînes de Markov. Les HMM sont très proches d'une représentation matricielle qui accepterait les *gaps* ([Hertz and Stormo, 1994]).

Ces représentations sont de type statistique. Leur construction nécessite un jeu d'apprentissage, souvent de taille conséquente. Si leur structure d'automate permet d'exprimer des branchements et donc des corrélations entre positions d'un même site, un tel modèle introduit un grand nombre de paramètres qu'il peut être difficile d'estimer. Ce type de représentation peut donc apporter des sensibilités et spécificités accrues au prix d'une complexité élevée. À ce titre, il se révèle être plus utilisé pour localiser un site connu que pour détecter un site inconnu.

Les réseaux neuronaux sont des graphes dont chaque noeud agit de manière comparable à celle d'un neurone. Une fonction de seuil détermine, en fonction des sommes des intensités des arcs entrants, la valeur à propager sur l'arc sortant. Ces représentations permettent d'exprimer des corrélations entre différentes positions d'un site ([Baldi and Brunak, 1998]). De même que les HMM, les réseaux neuronaux, structures complexes, nécessitent un jeu d'apprentissage pour leur construction. Mais ces derniers requièrent des jeux plus larges pour modéliser correctement un site, ce qui les rend inutilisables dans le cas (fréquent) où l'on dispose de peu d'exemples pour ce site.

2.3.4 Discussion

Nous avons décrit les principales représentations utilisées pour réunir des occurrences proches. Les motifs et matrices sont globalement les seules d'entre elles utilisées dans les problèmes d'inférence, et leur capacité de description des sites de fixation a souvent été jugée insuffisante. Ces représentations étant le lien qui nous mène de la biologie moléculaire à l'algorithmique, leur choix est cependant d'importance. À ce titre, elles limitent en effet les capacités de détection des algorithmes d'inférence à leur propre capacité de modélisation. C'est la raison pour laquelle, de notre opinion, toute recherche dans le domaine de l'inférence de sites de fixation doit, au vu des connaissances actuelles des caractéristiques de ces sites, passer par une étape de recherche d'une représentation apportant de meilleures capacités de description.

Étant donné qu'aucune de ces représentations ne s'est à ce jour avérée pleinement satisfaisante pour décrire correctement les particularités des sites de fixation, il semble qu'il reste de gros efforts de description à fournir pour mieux modéliser ces sites. Pour cela, une étude biologique des sites connus visant à mettre en évidence de nouvelles caractéristiques distinctives apparaît comme incontournable, impliquant une nécessaire collaboration entre biologistes moléculaires et informaticiens. Comme nous le verrons dans le chapitre qui suit, la tendance actuelle dans le domaine de l'inférence de sites de fixation s'inscrit dans cette logique d'affinement des représentations.

Il nous faut maintenant traiter des différents algorithmes d'inférence existants, qui utilisent les représentations décrites dans ce chapitre. Ces deux notions, représentation et algorithme, sont fortement liées ; les représentations de type déterministe comme les motifs consensus appartiennent à des espaces combinatoires dont les éléments sont aisément dénombrables. Dans le cadre d'une détection, ces espaces peuvent être décrits et parcourus sans difficultés par des algorithmes combinatoires, heuristiques, ou exacts si la taille de l'espace le permet ou si des propriétés peuvent être appliquées qui évitent de parcourir l'intégralité de cet espace. Les motifs consensus possèdent en effet des caractéristiques précises (la distance des occurrences avec un consensus) qui peuvent permettre de déduire des règles qui éviteront à un algorithme d'inférence de parcourir certaines parties de l'espace des solutions d'un problème donné. La représentation joue ici un rôle fort ; on dit que ces approches sont guidées par la représentation qu'elles utilisent ([Brazma et al., 1998a] emploie le terme de *pattern-driven*).

A contrario, on ne peut considérer la représentation matricielle indépendamment du problème posé. En effet, si l'ensemble des matrices possibles est infini et ne peut donc être parcouru, celui des alignements possibles dans un jeu de séquences donné est fini (et généralement plus vaste que l'ensemble des motifs consensus ayant au moins une occurrence dans le même jeu de séquences, sur un langage donné). Cependant, les motifs réunis dans ce type de représentation ne sont liés par aucune caractéristique précise. On ne peut donc pas déduire de propriétés intrinsèques qui éviteraient de parcourir l'intégralité de l'espace des solutions d'un problème, d'où l'utilisation nécessaire d'autres propriétés des sites de fixation (comme leur caractère exceptionnel) et l'emploi quasi-systématique d'heuristiques pour résoudre les problèmes utilisant ces représentations. Ces approches sont dites guidées par les séquences qu'elles traitent (*sequence-driven*).

Le compromis à faire entre la capacité de description d'une représentation et sa complexité, aperçu dans ce chapitre, était prévisible. La principale difficulté réside dans le fait qu'il semble nécessaire d'améliorer les représentations utilisées pour parvenir à de meilleures capacités de détection, mais que ces améliorations apportent des surcoûts qui rendent cette détection délicate à l'intérieur de jeux de données de grande taille que nous sommes souvent amenés à traiter dans des problèmes d'inférence. Nous verrons que la majorité des algorithmes d'inférence existants se limitent d'ailleurs au traitement de jeux de données assez réduits.

Partis d'un problème biologique, nous abordons le domaine de l'algorithmique et nous nous heurtons d'ores et déjà à un compromis à faire entre contraintes biologiques et algorithmiques. Faut-il privilégier une représentation plus fine des sites que nous recherchons ou bien l'assurance d'obtenir la meilleure solution à notre problème ?

Sur ces constats concernant l'indispensable affinement des représentations et la nécessité de pouvoir traiter de gros jeux de données, nos travaux prennent le parti d'utiliser des représentations similaires aux motifs consensus, pour leur caractère simple et le fait qu'ils nous permettent de développer des algorithmes exacts. Prenant note de leurs points faibles en matière de description des sites de fixation, nos premiers travaux consisteront à déterminer et à incorporer d'autres critères pertinents à cette représentation, afin de la rendre plus sensible et surtout spécifique, et ce sans augmenter excessivement sa complexité.

Dans le chapitre suivant, l'état de l'art des algorithmes d'inférence de sites particuliers va nous permettre de présenter les différentes approches du problème et de passer en revue les variantes de représentations utilisées à ce jour pour décrire au mieux les sites de fixation.

Chapitre 3

État de l'art de l'inférence de sites

Munis maintenant de plusieurs représentations informatiques pour des sites de fixation, nous évoluons dans le domaine de l'algorithmique. Ce chapitre est consacré à l'état de l'art dans le domaine de l'inférence de ces sites et de sites protéiques aux caractéristiques proches. Les algorithmes cités ici utilisent des représentations se basant sur celles que nous avons décrites précédemment. Nous prêterons une attention particulière aux modifications apportées à ces représentations dans certaines approches, visant à mieux modéliser les sites de fixation recherchés.

Les deux principales interrogations posées en bio-informatique au sujet des sites de fixation sont les suivantes :
- où se trouve un site donné dans une séquence donnée ? (problème de localisation ou recherche) ;
- peut-on détecter un site potentiel dans un jeu de séquences donné ? (problème d'inférence).

Les problèmes de localisation et d'inférence étant proches, nous donnons un aperçu du premier et des représentations et méthodes utilisées pour le résoudre dans la section 3.1. Nous verrons que les conditions de ce problème favorisent l'utilisation de représentations privilégiant les capacités de description du site, et que la complexité de ces représentations y a moins d'importance que dans un problème d'inférence.

Dans la section 3.2, nous nous attachons à définir d'un point de vue algorithmique les caractéristiques du problème d'inférence que nous nous proposons de résoudre. Nous présentons deux instances du problème d'inférence : celle de motifs diagnostics dans des séquences protéiques et celle qui nous intéresse. À partir de ces deux variantes, nous détaillons les spécificités de notre problème et commençons à cerner les différences conceptuelles qui existent entre les deux types d'approches utilisées à ce jour pour le résoudre : les méthodes d'optimisation et les combinatoires.

Dans la section 3.3, nous traitons des différentes façons d'évaluer la signification biologique d'un ensemble d'occurrences d'un site potentiel, indispensables aux algorithmes d'inférence.

Enfin, nous passons en revue dans la section 3.4 les principaux algorithmes d'inférence de sites existants, en distinguant les algorithmes d'optimisation des combinatoires.

L'état de l'art présenté dans ce chapitre est une version actualisée de celui qui est développé dans l'article de revue [Vanet et al., 1999].

3.1 Le problème proche de la localisation

Étant donné un site précis, décrit par une des représentations vues précédemment, et un jeu de N séquences $S_1 \dots S_N$, un problème de localisation consiste à trouver les positions des différentes occurrences de ce site dans chacune des séquences S_i.

Dans cette section, nous nous contenterons de passer en revue les principaux algorithmes de recherche de motifs approchés appliqués à la localisation de sites de fixation, en prêtant une attention particulière aux représentations employées.

3.1.1 Motifs et matrices

Le problème de localisation, mieux connu sous le nom de *pattern matching* quand on manipule des mots, a fait l'objet de nombreux travaux en algorithmique du texte depuis plus de trente ans. Le défi des approches de *pattern matching* consiste à réaliser la recherche dans un texte des positions (parfois seulement de la première d'entre elles) d'un mot donné le plus rapidement possible, c'est-à-dire en effectuant le moins d'opérations de comparaison possible. Le problème de localisation de sites biologiques dans un jeu de séquences ne se pose pas exactement dans les mêmes termes puisque, si les temps de recherche restent évidemment importants, s'y ajoutent des questions de sensibilité et spécificité propres aux sites recherchés.

La localisation de sites biologiques à l'aide de motifs est cependant assez proche des problèmes algorithmiques classiques de recherche de motifs approchés. Dans ce domaine, de nombreux travaux existent (dont on trouvera de plus amples références dans [Crochemore and Rytter, 1994] et [Gusfield, 1997]). Ce domaine a cependant connu un essor nouveau avec l'avènement de la bio-informatique. Si les algorithmes classiques de recherche de motifs approchés ont été directement adaptés à des problèmes de localisation dans des séquences biologiques (voir [El Mabrouk, 1996]), il est désormais convenu que, concernant la localisation de sites de fixation, les motifs consensus ne sont pas les plus aptes à décrire des sites de fixation. De plus, un des avantages des motifs consensus, leur simplicité, perd un peu de son intérêt dans des problèmes de localisation, beaucoup moins coûteux que ceux d'inférence.

La majorité des algorithmes de localisation de sites utilisent donc une représentation matricielle. Une matrice est construite à partir d'un jeu d'apprentissage (un ensemble d'occurrences du site que l'on souhaite localiser), et on l'utilise ensuite pour détecter le site recherché dans un jeu de séquences.

De nombreux algorithmes existent. Certains utilisent des profils, qui ne tiennent pas compte de la composition globale de la séquence (le contexte, *background* en anglais) d'où sont issus les sites ([Bucher and Bryan, 1984], [Mulligan et al., 1984], [Staden, 1984], [Schneider and Stormo, 1986], [O'Neill, 1989a], [O'Neill, 1989b], [O'Neill and Chiafari, 1989], [Frech et al., 1993a], [Frech et al., 1993b], [Hertz and Stormo, 1994]). D'autres algorithmes utilisent des matrices *log-likelihood*, qui prennent en compte ce contexte ([Berg and von Hippel, 1987], [Berg and von Hippel, 1988], [Stormo, 1988], [Stormo and Hartzell III, 1989], [Stormo, 1990], [Bucher, 1990], [Prestridge and Stormo, 1991], [Cardon and Stormo, 1992], [Lawrence et al., 1993], [Prestridge and Stormo, 1993], [Chen et al., 1995], [Hertz and Stormo, 1996]).

Parmi les différentes approches visant à améliorer la représentation construite à partir du jeu d'apprentissage, certaines permettent de s'affranchir du problème des jeux d'apprentissage trop réduits pour construire une matrice correcte ([Chen et al., 1998], [Harley and Reynolds, 1987]). D'autres, tels les algorithmes de Frech *et al.* prennent en compte le fait que le jeu d'apprentissage peut contenir des informations redondantes en éliminant les séquences promotrices qui sont trop proches les unes des autres ([Frech et al., 1993a], [Frech et al., 1993b]). Il a également été suggéré de pondérer chaque site du jeu d'apprentissage en fonction de son niveau d'expression ([Duret and Bucher, 1997]), mais ces données ne sont pas souvent disponibles.

En dépit d'améliorations significatives durant les dernières années, le taux de reconnaissance des outils de localisation reste assez faible : une étude faite par Duret *et al.* en 1997 sur les promoteurs eucaryotes (les performances sur les procaryotes doivent être du même niveau de fiabilité) constate que 30 à 40% des véritables promoteurs sont manqués, et que 45 à 60% des promoteurs localisés sont des *faux positifs* [1] ([Duret and Bucher, 1997]).

Hertz et Stormo, constatant les mêmes résultats médiocres, avancent plusieurs raisons à cela ([Hertz and Stormo, 1996]) :

- beaucoup de promoteurs doivent être associés à des sites d'activation pour être pleinement efficaces. Puisque toutes les méthodes actuelles d'apprentissage fabriquent une matrice à partir d'occurrences de sites sans se soucier d'éventuels sites d'activation liés, elles passeraient à côté d'éléments essentiels à la fonction du site qu'elles cherchent à détecter ;
- l'activité d'un promoteur n'est probablement pas additive le long de ses positions ;
- comme nous l'avons noté précédemment, beaucoup de facteurs physico-chimiques entrent en jeu dans la reconnaissance des sites de fixation, qu'aucune représentation ne prend en compte à ce jour.

On peut ajouter qu'aucune des méthodes actuelles, exceptées celles de Hertz et Stormo ([Hertz and Stormo, 1994]) et Frech *et al.* ([Frech et al., 1993a], [Frech et al., 1993b]) de façon limitée, ne prennent en compte de possibles corrélations entre les différentes posi-

[1] on appelle ***positifs*** et ***négatifs*** les résultats reportés et non reportés par un algorithme, et on les qualifie de *vrais* ou *faux* selon qu'ils sont effectivement des résultats corrects ou pas. Par exemple, un faux positif désigne un résultat qui a été reporté par un algorithme alors qu'il n'est pas correct. Un algorithme parfait ne produit pas de faux positifs ou de faux négatifs.

tions d'un site de fixation (comme Burge le fait pour les sites d'épissage [2] ([Burge, 1997], [Burge and Karlin, 1997], [Burge, 1998]) qui pourraient entraîner des calculs très coûteux. De même, peu d'algorithmes considèrent d'éventuelles corrélations entre plusieurs sites, c'est-à-dire des associations ordonnées et contraintes des divers éléments composant un promoteur, ou leurs positions relativement à d'autres éléments du gène, dans le but de réduire les faux positifs ([Prestridge and Stormo, 1991], [Prestridge and Stormo, 1993], [Prestridge, 1993], [Bucher et al., 1996], [Quandt et al., 1996], [Brazma et al., 1997], [Duret and Bucher, 1997], [Perez-Rueda et al., 1998], [Klingenhoff et al., 1999]).

3.1.2 Modèles de Markov et réseaux neuronaux

Des algorithmes de prédiction de gènes utilisant des chaînes de Markov et recherchant les biais de composition locaux dans des séquences ont été étendus afin d'incorporer la détection de sites de promotion ([Lukashin and Borodovsky, 1998], [Salzberg et al., 1999]). Ohler *et al.* ont aussi élaboré un algorithme utilisant des modèles de Markov pour détecter spécifiquement des régions de promoteurs eucaryotes ([Ohler et al., 1999]). Toutes les approches actuelles utilisant des HMM pour détecter les positions d'un site ont les mêmes limitations. En particulier, elles sont très sensibles au fait que le jeu de données soit trop bruité, ce qui est le cas quand on utilise de longues séquences plutôt que de petites régions pré-sélectionnées.

Les réseaux neuronaux utilisés pour localiser des sites de fixation ([Demeler and Zhou, 1991], [O'Neill and Chiafari, 1991], [Horton and Kanehisa, 1987]) font preuve d'une meilleure tolérance au bruit. Horton et Kanehisa ont montré que des différences significatives de sensibilité apparaissent entre représentations matricielles et réseaux neuronaux, en faveur de ces derniers, en particulier quand le jeu de séquences est suffisamment important et que le nombre de faux négatifs est modéré. Les réseaux neuronaux ont besoin de plus de données d'apprentissage pour effectuer de bonnes prédictions.

3.1.3 Discussion

Le problème de la localisation de sites, bien que proche, est beaucoup moins complexe que celui de leur inférence puisque dans ce cas on ne connaît pas le site recherché. Cette relative simplicité du problème de localisation a permis de concentrer les efforts des algorithmes actuels vers une recherche d'une meilleure sensibilité aux sites à localiser, la complexité de la représentation utilisée ayant une importance réduite.

Les représentations de base confirmant leur incapacité à décrire des sites de façon correcte, plusieurs améliorations ont été proposées, tenant compte de certaines spécificités du problème biologique, parmi lesquelles :

- prendre en compte des corrélations entre différentes positions d'un site. Le nombre maximum de substitutions d'un motif consensus est par exemple, de façon limitée, un facteur de corrélation : la substitution jouée à une position d'une occurrence ne

[2] ces signaux délimitent les introns et permettent l'épissage.

pourra être jouée ailleurs. Cependant, des corrélations plus complexes pourraient entrer en jeu dans la spécificité des sites ;
- considérer plusieurs sites non plus indépendamment les uns des autres, mais de façon coopérative. Les objets recherchés seraient alors des ensembles de sites ordonnés et associés sous certaines contraintes ;
- tenir compte de la position des sites relativement à d'autres éléments du gène.

Ces améliorations, bien qu'apportant nécessairement des surcoûts plus ou moins importants dans des algorithmes d'inférence, peuvent être des directions intéressantes à explorer dans le but d'améliorer leurs capacités de détection.

3.2 Spécificités du problème d'inférence de sites

Deux problèmes biologiques importants qui motivent l'élaboration d'algorithmes d'inférence sont (voir [Brazma et al., 1998]) :

1. La découverte de sites nucléiques dont on pense qu'ils sont impliquées dans la régulation de gènes d'un organisme donné. C'est le problème auquel nous nous intéressons.
2. Le besoin de détecter des *signatures* [3] (on parle aussi de motifs diagnostics) permettant de statuer sur l'appartenance d'une séquence protéique à une famille de protéines donnée.

D'autres problèmes biologiques tels la reconnaissance du site de fixation ribosomal dans des séquences d'ARNm (voir [Tompa, 1999] pour le détail du problème) ou de signaux particuliers (sites d'épissage, *enhancers*, sites de recombinaison ou de méthylation...) sont également du ressort des algorithmes d'inférence.

Nous décrivons plus en détail dans cette partie les deux problèmes d'inférence cités plus haut, ceci afin de cerner les spécificités de celui auquel nous nous intéressons et d'introduire la question de la validation des résultats obtenus.

3.2.1 L'inférence de sites de fixation : un problème de conservation

Parmi les propriétés des sites de fixation énumérées dans le chapitre 1, nous avons parlé d'une ressemblance, que nous avons utilisée pour définir des représentations dans le chapitre précédent. Cette ressemblance est supposée exister :
- entre les sites de gènes apparentés d'organismes différents ;
- entre les sites des gènes d'un même organisme qui sont co-régulés.

Quel que soit le cas de figure que l'on traite, on dispose d'un jeu de séquences dans lesquelles un site est censé apparaître sous forme d'occurrences proches. Dans le chapitre 1, nous relevions également que ce site doit être un « îlot » fonctionnel (puisqu'il initie la mise

[3] une signature est un motif propre à une famille de séquences protéiques données, et qui permet donc de la distinguer.

en place du mécanisme de transcription) au milieu du bruit de la séquence non-codante, et doit donc avoir un caractère « exceptionnel » (ou « inattendu », termes définis et discutés dans la section suivante), par rapport à l'ensemble des séquences.

Nous nous intéressons ici au cas où l'on dispose de gènes d'un même organisme, et où l'on suppose que des gènes co-régulés y ont des promoteurs et/ou sites de régulation communs. Ce cas est le plus général, car, quand on suppose une conservation entre gènes ayant divergé, d'autres données doivent être prises en compte qui ne sont pas toujours disponibles, comme des distances d'évolution entre chacun des organismes considérés. En effet, le principe de conservation des sites de fixation entre gènes ayant divergé s'appuie sur l'hypothèse que ces sites actifs évoluent moins vite que le non-codant à cause de la pression de sélection qu'ils subissent, ce qui permet de les mettre en évidence. Cette approche (dite « des empreintes phylogénétiques ») nécessite de disposer de *gènes orthologues* [4], puisqu'elle utilise des distances évolutives pour mesurer les similarités. Ce type d'approche nécessite également que ces gènes soient assez éloignés d'un point de vue évolutif pour que les séquences non-codantes qui sont en amont aient divergé différemment, mais assez proches pour que les sites eux-mêmes n'aient pas trop divergé, auquel cas ils ne pourront plus être apparentés. Les conditions requises sont donc assez restrictives. Nos études et recherches se concentrent sur le cas plus général, où l'on ne dispose que des séquences non-codantes d'un organisme donné, sans autre information. On trouvera dans [Duret and Bucher, 1997] et [Tompa, 2001] un aperçu des approches phylogénétiques existantes.

Nous considérons donc que nous disposons de séquences d'un même organisme, et supposons que les gènes co-régulés partagent certains sites de fixation. Les annotations de certains gènes font partie des rares données qui sont souvent disponibles concernant les séquences des banques. Elles indiquent où se trouvent certains gènes. Considérant donc que l'on dispose de tout ou partie des séquences non-codantes d'un génome, il s'agit donc de détecter des occurrences proches apparaissant dans plusieurs de ces séquences et ayant un caractère « exceptionnel ».

On ne sait pas *a priori* dans quelle proportion des séquences un même site peu apparaître.

Définition 3 (Couverture d'un site)
La couverture en séquences *d'un site représente le nombre de séquences dans lesquelles des occurrences de ce site apparaissent au moins une fois. Sa* couverture en occurrences *correspond au nombre total d'occurrences de ce site.*

En nous plaçant dans le contexte où nous disposons de séquences prélevées en amont de plusieurs gènes (ce qui est envisageable pour des séquences annotées), c'est la couverture en séquences du site qui peut s'y trouver qui nous intéresse. Sauf mention contraire, le terme couverture désignera donc cette couverture en séquences.

---

[4] on dit de gènes ayant un ancêtre commun qu'ils sont homologues. S'ils en sont issus par spéciation ce sont des orthologues, si c'est par duplication on parle de paralogues.

La couverture du site est d'une grande importance pour les algorithmes d'inférence dont nous traitons. Ce critère de couverture est propre au site, dépendant du nombre de gènes régulés par une même famille de promoteurs ou de sites de régulation dans un jeu de données, mais dépend également de ce jeu de données. Si le jeu de séquences à partir duquel on infère un site est issu d'expériences biologiques (paillasse, ou puces à ADN [5]) ayant permis d'isoler un jeu de séquences dites « propres », dans lesquelles on suppose que le site est très représenté, on peut espérer une couverture importante. Dans le cas brut où le jeu est simplement constitué de séquences non-codantes d'un génome, réunies sans autre raison que leur appartenance à un organisme commun, la couverture peut être très faible.

Un algorithme d'inférence ayant pour but de résoudre un problème de conservation dans n'importe quelles conditions doit être capable de gérer une couverture de tout ordre, et *a priori* inconnue.

Si on considère que la présence de sites communs à des gènes co-régulés n'est pas assurée, nous disposons, au final, d'assez peu de propriétés permettant de détecter les sites de fixation. C'est pourquoi les algorithmes d'inférence se basent en général sur le seul critère commun qu'on leur connaît à ce jour : leur caractère « exceptionnel ». C'est la mesure de *signification* d'un site (qui peut être évaluée de différentes façons, comme nous le verrons dans la section 3.3) qui permet de statuer sur ce caractère « exceptionnel ». Elle donne une estimation de la valeur fonctionnelle d'un site, indiquant si sa présence et sa couverture sont dues au hasard (à la composition des séquences) ou pas.

3.2.2 L'inférence de signatures : un problème de classification

Dans cette instance du problème d'inférence, on dispose par exemple d'une famille de séquences protéiques (regroupées selon un caractère fonctionnel ou physico-chimique qu'elles partagent) et d'un ensemble d'autres séquences protéiques (le plus souvent une banque complète) n'appartenant pas à cette famille.

On souhaite inférer de la famille de séquences un motif diagnostic, qui permettra de statuer sur l'appartenance d'une séquence protéique quelconque à cette famille. On cherche donc un motif qui apparaisse dans toutes les séquences de la famille fournie et dans aucune des séquences de la banque dont on dispose. Évidemment, ce cas est idéal, et l'on devra prendre en compte la présence éventuelle de séquences étrangères à la famille protéique dans le jeu initial, ainsi que celle de séquences appartenant à la famille protéique dans la banque. C'est un problème de classification.

Cette instance du problème d'inférence, bien que proche, diffère de la précédente car

[5] les puces à ADN permettent de mesurer le niveau d'expression d'un groupe de gènes dans des conditions particulières. Le fait que des gènes co-exprimés partagent les mêmes mécanismes de régulation, bien qu'étant une hypothèse raisonnable, n'a pas été établi à ce jour, des mécanismes différents pouvant produire le même profil d'expression. Sous cette hypothèse, les données issues de puces à ADN seraient d'intérêt pour des algorithmes d'inférence de sites de fixation, permettant de disposer de jeux de séquences dans lesquels un site aurait une grande couverture et facilitant de ce fait grandement sa détection.

on dispose ici d'un jeu de contre-exemples (la banque) qui permet de valider ou d'invalider de potentiels motifs diagnostics. Il n'est donc pas nécessaire d'avoir recours à une mesure de signification. Ce problème diffère également du précédent sur le point de la couverture : les motifs à extraire auront (par définition) une forte couverture du jeu initial.

Notons que le problème d'inférence de sites de fixation vu précédemment peut aussi être posé comme un problème de classification quand on dispose d'un jeu de séquences dans lesquelles on sait que le site recherché a peu ou pas de chances d'apparaître. On peut alors utiliser ce jeu de comparaison pour valider ou invalider des sites potentiels.

Inversement, le problème de classification de séquences protéiques apparaît comme un sous-problème de celui que nous nous posons. Un algorithme qui résoud de façon exacte un problème de détection de sites de fixation pourra aisément être transposé à ce type de problèmes.

3.2.3 Discussion

Ces deux instances du problème d'inférence mettent en avant les deux types de critères selon lesquels un site est généralement inféré :

- les critères structurels du site, longueur, couverture, conservation, corrélations internes ou avec d'autres sites. Ces critères sont spécifiés à l'avance ou évalués lors de l'inférence ;
- les critères de signification, qui imposent aux sites inférés d'avoir un caractère « exceptionnel ». Nous traitons de ces critères dans la section qui suit.

Parmi les critères structurels, la couverture joue un rôle particulier. Elle dépend du site mais aussi du jeu de données utilisé.

Par rapport à un problème de classification, un problème de conservation a pour spécificité de nécessiter l'emploi de critères de signification. Ces critères ont un rôle important lors de l'inférence puisqu'ils sont les seuls à permettre d'estimer la validité d'un site. On peut d'ores et déjà esquisser plusieurs tactiques d'inférence de sites de fixation privilégiant l'une ou l'autre des deux classes de critères :

1. Se baser sur des critères structurels connus à l'avance pour réduire l'espace des solutions et extraire des sites dont on évaluera ensuite la signification.
2. Se baser sur les critères de signification pour évaluer les meilleurs critères structurels.
3. Un mélange des deux approches précédentes consistant à extraire des sites sur des critères structurels donnés et sur des critères de signification simultanément, de façon à diminuer l'espace des solutions à visiter.

Il existe deux grandes approches utilisées par les algorithmes traitant le problème d'inférence de sites : les méthodes d'optimisation et les combinatoires. Les algorithmes combinatoires utilisent en général la première ou la troisième tactique pour inférer un site de fixation. Les algorithmes d'optimisation utilisent la seconde, cherchant, comme leur nom l'indique, à optimiser les critères de signification pour extraire le jeu d'occurrences ayant le caractère le plus exceptionnel.

Quelle que soit l'approche utilisée, on ne doit pas perdre de vue que, faute de disposer de connaissances suffisantes sur les caractéristiques des sites de fixation, les résultats obtenus quand on infère ces sites sont entièrement conditionnés par la mesure de signification choisie, dont nous traitons dans la section suivante, puisqu'on utilise toujours un modèle d'estimation dont on suppose théoriquement ou empiriquement qu'il est proche de la réalité.

Il est, en ce sens, délicat de parler d'algorithmes exacts. Nous appliquons ce qualificatif en fonction des critères principaux utilisés pour réaliser l'inférence. Si ces critères sont structurels, l'algorithme d'inférence peut être de type exact. Si l'algorihme est basé sur des critères de signification, la complexité du problème d'inférence posé en ces termes impose l'utilisation d'heuristiques pour le résoudre.

Rappelons que la majorité des algorithmes d'inférence perçoivent le problème sous un angle particulier. On ne considère en effet que des séquences primaires, aucune information de configuration spatiale des macro-molécules utilisées n'est prise en compte. Sauf mention particulière, les algorithmes décrits dans ce chapitre utilisent uniquement des séquences non-alignées, sous forme de texte sur un alphabet fini.

3.3 Mesurer la pertinence d'un site

Les connaissances actuelles en matière de sites de fixation font état de peu de caractéristiques générales propres à ces sites. Chez certains organismes procaryotes, des caractéristiques structurelles particulières ont été observées, comme l'association entre eux de certains sites due à la conformation spatiale du ou des enzymes qui viennent s'y fixer (comme dans l'exemple des facteurs σ^{70} de *E.coli*, figure 1.3 page 27). Ces associations ne sont pas pour autant observables systématiquement, et ne peuvent donc pas être imposées lors de l'inférence de n'importe quels sites de fixation. Les critères structurels connus de ces sites sont en général spécifiques à une famille d'organismes précise. En outre, seuls, ils sont souvent insuffisants pour permettre d'isoler de tels sites.

La recherche d'une solution au problème de la détection de sites de fixation s'appuie nécessairement sur une des rares caractéristiques qui leur soit connue et commune : leur caractère « exceptionnel ». Du fait de leur caractère fonctionnel, ces sites sont en effet sensés apparaître de façon exceptionnellement fréquente en amont des gènes, et de façon exceptionnellement rare ailleurs.

C'est pourquoi des critères de signification sont nécessaires quand on infère des sites de fixation. Ces critères sont généralement exprimés par le biais d'une mesure de signification permettant de quantifier le caractère exceptionnel de ces sites.

Quand on infère des sites de fixation, on doit donc faire le choix d'une méthode de mesure pour estimer la signification d'un site. L'enzyme qui vient se fixer sur un promoteur plutôt qu'à une autre position de la séquence a implicitement respecté, à travers des propriétés physico-chimiques, une méthode de mesure qui a privilégié cette position. De même un aimant se trouvant à proximité de plusieurs objets respectera des lois physiques pour venir éventuellement se fixer sur l'un des objets. Mais si les lois qui gouvernent cet aimant et le contexte dans lequel il évolue sont à ce jour bien connues, permettant de prédire avec précision quelle sera la trajectoire de ce dernier, il est loin d'en être de même pour les sites de fixation. Quand bien même les forces qui régissent les interactions ADN/protéines seraient connues, la complexité de l'environnement à prendre en compte lors de ces interactions rend le problème probablement trop complexe pour être résolu comme celui de la trajectoire d'un simple aimant. Aussi, les méthodes de mesure que nous choisirons se basent elles sur des hypothèses.

Le côté arbitraire de cette mesure est encore renforcé par la notion de contexte, c'est-à-dire tout ce qui, dans la séquence, n'est pas le signal qui nous intéresse. C'est en effet par rapport à ce contexte que l'on va évaluer le caractère exceptionnel d'un site. Comme nous le disions lors de l'énumération des propriétés des sites de fixation au terme du chapitre 1, nous ne devons pas dissocier leur caractère exceptionnel de la question « exceptionnel par rapport à quoi ? », question qui reste à ce jour sans réponse précise. Si un site de fixation ne doit effectivement pas apparaître fréquemment, au risque d'induire des transcriptions erronées, comment juger du « domaine de visibilité » des facteurs qui les reconnaissent ? Certains auteurs d'algorithmes d'inférence ([Roth et al., 1998]) émettent par exemple l'hypothèse (pour laquelle des contre-exemples existent) qu'un site fonctionnel ne doit pas (ou très peu) apparaître dans des séquences codantes.

De plus, s'il est probable qu'un ensemble de sites de fixation (promoteur et sites de régulation) ne doivent pas apparaître n'importe ou, qu'en est-il d'un site considéré isolément ?

Pour toutes ces raisons, les mesures de signification, si elles se proposent de traiter un des seuls critères propres aux sites de fixation, sont à manipuler avec beaucoup de précautions. Il est important de garder à l'esprit qu'elles se fondent toujours sur des hypothèses que nous ne pouvons vérifier à ce jour.

Cette section n'a pas pour ambition de décrire l'ensemble des méthodes existantes qui permettent d'estimer la signification d'un site (voir [Brazma et al., 1998a] pour un aperçu), mais seulement d'introduire les bases nécessaires à l'explication des mesures utilisées dans la majorité des algorithmes, dont les nôtres.

3.3.1 Comptage simple

Une hypothèse simple pour estimer la signification d'un site consisterait à dire que son nombre d'occurrences dans un jeu de séquences est en soi une mesure de sa signification. Expliquons pourquoi un site ayant un grand nombre d'occurrences n'a pas nécessairement un caractère exceptionnel.

Il est assez courant que des génomes aient un biais de composition plus ou moins fort, par exemple que le taux de AT soit significativement plus élevé que celui de GC ou inversement [6]. Dans un génome AT-riche, si l'on extrait les motifs les plus répétés, on a plus de chance d'obtenir des motifs riches en AT. Ces derniers auront donc un caractère moins exceptionnel que des motifs riches en GC, dans ce cas.

Dans le jeu de séquences de la figure 3.1, les séquences ont un fort biais en AT ($A = 51\%$, $C = 8\%, G = 3\%, T = 38\%$). Si on se contente de compter les trinucléotides les plus fréquents, on obtient le classement suivant :
- sans substitution : $ATA = 20, TAT = 14, CAA = 2, TCA = 2$ et le reste à 1 ;
- avec une substitution : $ATA = 25, TAT = 19, TCG = 5, CAA = 4, TCA = 4, \ldots$

Ces résultats ne permettent pas de mettre en évidence que le motif TCG^1 (avec une substitution) est exceptionnellement présent : il apparaît pourtant dans cinq séquences sur six bien qu'il soit composé de deux symboles (C et G) très faiblement présents dans les séquences.

$\text{ATATA}\underline{\text{GCG}}\text{ATA}$
$\text{ATATA}\underline{\text{TCT}}\text{ATA}$
$\text{ATATA}\underline{\text{TCA}}\text{ATA}$
ATATATATATA
$\text{ATATA}\underline{\text{TCA}}\text{ATA}$
$\text{ATATA}\underline{\text{TCC}}\text{ATA}$

FIG. 3.1 – Comptage dans un jeu de séquences nucléotidiques riche en AT.
Composition : $A = 51\%, C = 8\%, G = 38\%, T = 3\%$. Le motif TCG avec une substitution (souligné), exceptionnellement présent en regard du biais en AT de la séquence.

Pour mesurer le caractère exceptionnel d'un site, il est donc nécessaire de prendre en compte la composition des séquences dont il est extrait. Notons que la façon dont on relève la composition du jeu de séquences est primordiale : en ne considèrant que la composition en simples nucléotides (comme c'est le cas quand on qualifie une séquence d'AT- ou GC-riche), un motif ATA pourra difficilement être considéré comme exceptionnellement représenté dans une séquence AT-riche, alors qu'il pourrait très bien l'être si on considère la composition en di-nucléotides de la séquence.

[6] les deux brins d'ADN étant complémentaires, il y a dans un génome donné autant de A que de T, et autant de G que de C. On parle donc de composition en AT ou en GC. Un génome ou un ensemble de séquences dont la fréquence de AT est significativement élevée est dit AT-riche.

3.3.2 Entropie relative

Nombre d'algorithmes utilisent la mesure d'entropie relative (ou divergence de Kullback-Leibler, [Kullback and Leibler, 1951]) afin de prendre en compte la composition de la séquence pour évaluer le caractère exceptionnel d'un site donné. L'entropie relative d'un motif nucléique m de longueur k sur l'alphabet Σ s'obtient par la formule :

$$\sum_{i=1}^{k} \sum_{\alpha \in \Sigma} f_{\alpha,i} \log_2 \frac{f_{\alpha,i}}{f_\alpha}$$

où $f_{\alpha,i}$ est la fréquence avec laquelle on observe le symbole α à la position i des occurrences de m, et f_α est la fréquence globale du symbole α dans l'ensemble des séquences privées des occurrences de m (le contexte).

Les matrices *log-likelihood* présentées dans le chapitre précédent sont particulièrement adaptées à l'emploi de cette mesure d'entropie relative, puisqu'il suffit de pondérer chaque valeur d'une telle matrice par la fréquence associée et de sommer les valeurs adéquates pour obtenir l'entropie relative d'une matrice donnée. Les algorithmes qui mesurent la signification d'un site grâce à l'entropie relative utilisent donc en général une représentation matricielle, et inversement. C'est souvent l'utilisation de cette entropie relative ou d'une autre mesure estimant le caractère exceptionnel des sites modélisés qui permet de guider les algorithmes heuristiques utilisant la représentation matricielle dans le vaste espace à visiter.

L'entropie relative mesure le caractère insolite d'un motif en regard de la composition globale en nucléotides d'une ou plusieurs séquences. Plus elle est élevée, plus le motif est considéré comme exceptionnel.

Cependant, l'entropie relative n'offre une bonne mesure de comparaison qu'entre des motifs ayant le même nombre d'occurrences. L'entropie relative de deux motifs qui ont des jeux d'occurrences de tailles différentes n'est pas comparable puisque seules les fréquences d'apparition des nucléotides entrent en jeu dans leur calcul, et que le nombre d'occurrences n'est pas pris en compte.

En reprenant l'exemple de la figure 3.1 et en calculant les entropies relatives de chaque trinucléotide, on obtient les résultats qui suivent (rappelons qu'on note AAA^1 le motif AAA avec une substitution, et notons $\mathcal{E}(AAA^1)$ l'entropie de ce motif) : $\mathcal{E}(TCA^0) = 6.08$, $\mathcal{E}(TCG^1) = 5.69$, $\mathcal{E}(TCA^1) = 5.38$, $\mathcal{E}(ATA^0) = 3.31$, $\mathcal{E}(ATA^1) = 2.98$. On constate d'abord que l'entropie relative prend bien en compte le biais en AT du jeu de séquences : le motif ATA obtient un faible score, avec ou sans substitution. Bien que les motifs TCA^0 et TCG^1 aient un score logiquement plus élevé que ce dernier, on constate cependant que TCA^0, qui a deux occurrences exactes parmi les six séquences, obtient un meilleur score que TCG^1, qui possède cinq occurrences. La différence, faible dans cet exemple, peut devenir importante quand les jeux d'occurrences ont des ordres de grandeur très différents. L'entropie relative, ne prenant pas en compte le nombre d'occurrences d'un site, peut donner un meilleur score à un site parfaitement conservé sur peu de séquences qu'à un site moins bien conservé qui couvre la quasi-totalité des séquences.

De ce fait, la plupart des algorithmes qui utilisent cette mesure supposent une couverture complète (ou presque) des séquences par les sites à inférer, ce qui est rarement le cas en pratique.

Une bonne mesure doit donc tenir compte de la composition du jeu de séquences et du nombre d'occurrences des sites. Van Helden *et al.* ont par exemple décrit des mesures prenant en compte ces deux paramètres, mais elles ne concernent que des motifs exacts([van Helden et al., 1998]).

Il est à noter que la validité de cette méthode de mesure de signification est discutée sur le point suivant : elle suppose que les entropies de chaque position d'une matrice s'additionnent pour donner l'entropie de cette matrice. Or rien ne prouve qu'il en va de même dans la réalité, et que l'activité d'un site dépend d'une relation additive sur ses positions.

3.3.3 Signification statistique

3.3.3.1 Z-score

Une autre façon d'évaluer la pertinence d'un site consiste à comparer sa probabilité d'apparition avec la fréquence d'apparition observée dans le jeu original.

En supposant que l'on peut estimer la probabilité p_m d'apparition d'un motif m donné, le *z-score* ([Alder and Roessler, 1972]) indique si le nombre d'occurrences N_m observé parmi les N séquences est exceptionnel ou non. Le z-score z_m d'un motif m vaut :

$$z_m = \frac{N_m - Np_m}{\sqrt{\sigma_m}}$$

où l'on considère, sous l'hypothèse que les N séquences sont indépendantes, que :

- Np_m est le nombre de séquences dans lesquelles on espère au moins une occurrence de m ;
- σ_m est l'écart-type associé à Np_m.

Un z-score positif indique que le modèle m est sur-représenté par rapport à sa probabilité d'apparition, un z-score négatif indique qu'il est sous-représenté. Enfin, un z-score nul indique que les résultats observés pour m correspondent aux résultats attendus.

Le z-score est une mesure qui prend en compte la composition de la séquence (par le biais de la probabilité d'apparition du motif) et le nombre d'occurrences observé. Elle respecte donc les deux critères qui nous semblent importants. De plus, elle est asymptotiquement normalement distribuée, et normalisée pour obtenir une moyenne de 0 et un écart-type de 1. Elle permet donc de comparer des sites de longueurs différentes.

3.3.3.2 χ^2

Le test du χ^2 (prononcer "Ki deux") est une autre mesure statistique permettant d'estimer le caractère exceptionnel d'un résultat par rapport à un autre. Supposons qu'on dispose des résultats suivants sur deux jeux A et B contenant respectivement N_A et N_B séquences : le motif m apparaît dans v_A séquences du jeu A et v_B séquences du jeu B. Soit

$\overline{v_A} = N_A - v_A$, $\overline{v_B} = N_B - v_B$, $n_V = v_A + v_B$ et $\overline{n_V} = \overline{v_A} + \overline{v_B}$. Le test du χ^2 avec un degré de liberté est défini par la formule suivante :

$$\chi^2 = \frac{(N_A+N_B)(v_A\times\overline{v_B}-v_B\times\overline{v_A})^2}{n_V\times\overline{n_V}\times N_A\times N_B}$$

En présence d'un jeu de résultats observés et d'un jeu de résultats attendus, le test du χ^2 produit une mesure de cohérence entre ces résultats d'origine différente. Plus la valeur obtenue est proche de 0, plus la cohérence entre elles est grande.

Ce test n'a cependant pas les propriétés de normalisation du z-score : le nombre d'occurrences du motif influe sur la valeur de son χ^2, indépendamment de la proportion d'occurrences apparaissant dans les deux jeux.

Les tests du z-score et du χ^2 nécessitent tout deux de se référer à un résultat attendu, une probabilité d'apparition. Une difficulté importante consiste à obtenir une valeur correcte pour cette probabilité d'apparition d'un motif et une évaluation juste de son écart-type (pour le z-score). La méthode la plus grossière consiste à utiliser un modèle de Bernoulli (chaque symbole a une probabilité d'apparition indépendante calculée directement à partir de la composition de la séquence). Cette probabilité peut également être calculée par une chaîne de Markov générée à partir du jeu de séquences. Disposant de cette probabilité p_m d'apparition d'un modèle, l'écart-type est souvent approché par la formule $\sigma = Np_m(1-p_m)$ (où N est le nombre de séquences).

Les travaux de Mireille Régnier, de l'INRIA, avec qui nous avons collaboré, concernent le calcul de la probabilité d'apparition de la première occurrence d'un motif dans une séquence, qui est celle qui nous intéresse. Elle est plus délicate et coûteuse à évaluer que la probabilité d'apparition classique, et ne dépend pas uniquement de la variance et de l'espérance du motif ([Régnier and Szpankowski, 1997]).

Nous avons également collaboré avec Jean-Jacques Daudin, Hugues Richard, Stéphane Robin et Sophie Schbath, de l'INA P-G, qui s'intéressent aussi à l'évaluation de cette probabilité particulière. Les méthodes qu'ils ont développées sont décrites dans [Robin et al., 2001].

Pour avoir un aperçu des nombreux travaux dans ce domaine, nous invitons le lecteur à se référer à la thèse récente de Lonardi ([Lonardi, 2001]) et à [Reinert et al., 2000].

Enfin, une autre façon d'obtenir une probabilité d'apparition d'un motif consiste à utiliser une méthode de simulation. On observera les fréquences d'apparition du motif dans des séquences générées aléatoirement et conservant la composition en k-mers [7] du jeu original (*shuffling*), et ce un nombre de fois suffisant pour obtenir des résultats stables. Faute de mieux, cette méthode assez coûteuse s'avère être des plus fiables.

Nous décrirons plus en détail dans la section 6.1.3 page 150 les critères de signification statistique que nous avons utilisés.

[7] k-mers ou k-tuples : mots de longueur k.

3.4 Les méthodes d'inférence existantes

Nous distinguons dans cette partie les méthodes d'optimisation des méthodes combinatoires ([Ohler and Niemann, 2001] parlent de méthodes d'alignement, et de méthodes énumératives ou exhaustives, [Brazma et al., 1998a] de méthodes guidées par les séquences ou guidées par les motifs).

3.4.1 Méthodes d'optimisation

De par la complexité du problème quand il est posé en termes d'optimisation des critères de signification, toutes les méthodes d'optimisation existantes sont des heuristiques.

3.4.1.1 EM (Expectation-Maximization) et échantillonnage de Gibbs

La méthode « *expectation-maximization* » (abregée EM) est une procédure en deux étapes qui cherche à établir un consensus (sous forme de matrice) ayant une forte (sinon la plus forte) signification, exprimée par une mesure d'entropie relative. Cette stratégie est conçue à l'origine pour inférer des sites d'une longueur donnée et ayant une couverture de l'ensemble des séquences ([Schneider and Stormo, 1986], [Stormo, 1988], [Stormo and Hartzell III, 1989], [Lawrence and Reilly, 1990], [Stormo, 1990], [Cardon and Stormo, 1992], [Hertz and Stormo, 1996]).

Les deux étapes d'un algorithme EM sont :

- *expectation* : on calcule l'entropie relative de l'alignement courant (le choix du premier alignement est primordial, certaines approches le choisissent aléatoirement, d'autres font un parcours d'une partie restreinte de l'espace des solutions à la recherche du meilleur alignement de départ) ;
- *maximization* : on cherche à maximiser cette valeur en modifiant l'alignement selon une stratégie précise.

Ces deux étapes sont répétées jusqu'à ce que l'on obtienne un alignement ayant une valeur maximale ou qu'on ait atteint un nombre maximum d'itérations. Cette approche est déterministe : pour un alignement initial donné on obtient toujours le même déroulement de l'algorithme et le même résultat.

Selon l'alignement initial choisi et la façon dont se déroule l'inférence, un algorithme EM peut-être amené à ne pas converger ou à converger vers un maximum local. En effet, on ne peut s'extraire d'un maximum local qu'en renonçant momentanément à optimiser l'entropie relative de l'alignement courant, ce que ne prévoit pas cet algorithme.

Ce problème trouve une solution possible dans l'approche du "Gibbs *sampling*" ([Lawrence et al., 1993]) où le choix d'un nouvel alignement ne se fait plus de façon optimale mais probabiliste : pour un alignement donné, on choisit une des séquences qu'on retire du calcul de l'alignement courant, puis on pondère chaque position de cette séquence avec le score qu'elle obtient en l'alignant avec l'alignement courant. Enfin, on sélectionne aléatoirement une de ces positions avec une probabilité qui dépend de la pondération donnée précédemment à chacune d'entre elles. Cette procédure est répétée pour chacune des séquences (séquentiellement ou de manière aléatoire) jusqu'à convergence. Les chances d'atteindre le maximum global augmentent à chaque itération. Lawrence *et al.* prouvent que l'algorithme converge, mais il n'atteint pas toujours le maximum global. Il a cependant moins de chance de se retrouver bloqué dans un maximum local qu'une approche de type EM. Cette approche est en fait une variante stochastique de la méthode EM : des « chemins » différents seront empruntés à chaque exécution sur un même jeu de données, et le résultat obtenu ne sera donc pas nécessairement le même.

Ces deux approches d'optimisation fournissent, au terme d'un cycle complet, une unique matrice correspondant au meilleur consensus détecté. Pour obtenir d'autres consensus intéressants, elles doivent réitérer le processus de convergence en évitant les positions déjà utilisées. En pratique, les temps d'exécution sont fortement alourdis par l'augmentation du nombre de consensus à produire.

La méthode de Roth *et al.* ([Roth et al., 1998]) est une variante de celle de Lawrence qui ne retient que les motifs qui ont moins de 1% de leurs occurrences dans un *ORF* [8]. Elle se base sur l'hypothèse que les sites de fixation, de par leur caractère exceptionnel, ne doivent pas (ou très peu) apparaître dans des ORF. Cette hypothèse n'a cependant jamais été validée et il existe des contre-exemples qui laissent à penser qu'elle n'est pas toujours juste.

Parmi les variantes du même genre, certains algorithmes basés sur l'approche EM ([Cardon and Stormo, 1992], [Frech et al., 1993a], [Frech et al., 1993b]) ajoutent aux paramètres de position sur lesquels la maximisation est effectuée des critères supplémentaires à estimer par l'algorithme, comme la longueur d'un *spacer* se trouvant à l'intérieur du consensus. Les paramètres à estimer, déjà nombreux dans ce type d'approche, se voient encore augmentés et peuvent donc donner lieu à des calculs très lourds.

Toutes ces méthodes considèrent, pour des raisons combinatoires et statistiques, que chaque séquence contient un et un seul site, de longueur fixée. Comme vu dans la section précédente, cela vient du fait que les mesures de signification d'un site utilisant l'entropie relative ne prennent pas en compte le nombre total d'occurrences mais seulement la fréquence relative d'apparition de chaque nucléotide d'une occurrence. Sous un modèle de distribution uniforme, un site parfaitement conservé apparaissant dans quelques séquences obtiendrait une meilleure probabilité d'apparition qu'un site mal conservé apparaissant dans la quasi-totalité des séquences.

Ces deux impératifs d'existence et d'unicité d'un site dans chacune des séquences sont résolus dans l'algorithme de Bailey ([Bailey and Elkan, 1994], [Bailey, 1995], [Bailey and Elkan, 1995], [Bailey and Elkan, 1995a]), implanté dans un outil appelé MEME, ainsi que dans d'autres approches de type Gibbs ou EM ([Rocke and Tompa, 1998], [Hu et al., 2000], [Rocke, 2000] qui autorisent les *gaps*). MEME utilise un algorithme de type EM avec un nombre plus important de paramètres à estimer (le nombre de sites par séquence, la cou-

[8]Open Reading Frame, zone susceptible d'être traduite, en y incluant les introns.

verture du site, sa longueur). La taille de l'espace des solutions s'en voit multipliée. Son coût est donc plus élevé que ceux des algorithmes de Stormo et Lawrence : ce dernier a un comportement moyen linéaire en la longueur des séquences traitées, tandis que MEME est quadratique en fonction de cette longueur. L'un des principaux atouts de MEME réside dans le peu d'informations que l'utilisateur doit fournir, puisque l'algorithme se charge d'estimer, à l'aide des critères de signification, les meilleurs critères structurels. MEME pâtit du même inconvénient que toutes les approches du type EM, à savoir qu'il peut converger vers un maximum local, et qu'il ne peut produire qu'un petit nombre de consensus.

L'algorithme « glouton » de Hertz *et al.* adopte une autre stratégie (déterministe) d'optimisation du score d'une matrice : parmi N séquences, le motif (de longueur fixée) de la première séquence qui obtient le meilleur score est aligné avec tous ceux de la seconde séquence. Parmi les alignements produits, la matrice obtenant le meilleur score est à son tour conservée pour être alignée avec les motifs de la troisième, et ainsi de suite jusqu'à ce qu'un certain nombre de motifs aient été alignés ou bien qu'un seuil de score ait été atteint ([Hertz and Stormo, 1999]).

Enfin, l'approche récente exposée dans [Thijs et al., 2001a] et [Thijs et al., 2001b], et implantée dans le logiciel Motifs Sampler, est une variante du "Gibbs *sampling*" : l'estimation du nombre de sites par séquence est faite par une distribution de probabilités, et le contexte est modélisé par une chaîne de Markov d'ordre quelconque (alors que la plupart des outils d'optimisation disponibles ne prennent en compte que la fréquence des nucléotides). Très récemment, [Liu et al., 2001b] ont présenté une approche similaire utilisant également des chaînes de Markov (d'ordre zéro à trois), et visant à mettre en évidence des paires de sites associés. [Guha-Thakurta and Stormo, 2001] ont également développé une variante du "Gibbs *sampling*" ayant le même objectif.

3.4.1.2 HMM

Les modèles de Markov cachés sont en général utilisés pour modéliser le consensus d'un site à partir d'un jeu de sites connus contenant éventuellement du bruit. Il n'existe à notre connaissance que peu d'algorithmes d'inférence utilisant des HMM. Celui de Yada n'utilise une représentation sous forme de HMM que dans sa seconde partie ([Yada et al., 1998]). Des oligonucléotides sont d'abord extraits selon un critère de signification basé sur un test du χ^2. Les oligonucléotides les plus significatifs et qui se recouvrent sont assemblés entre eux et un HMM est alors construit à partir de chaque assemblage. Ces HMM sont ensuite utilisés pour localiser les positions les plus significatives dans les séquences originales. Ces positions sont elles-mêmes réutilisées pour construire de nouveaux HMM, et ce jusqu'à convergence.

3.4.2 Méthodes combinatoires

Les méthodes combinatoires, comme celles d'optimisation, utilisent des critères de signification afin d'estimer le caractère exceptionnel d'un motif. Mais, à la différence de celles-ci, elles ne sont pas nécessairement guidées par ces critères. Ceux-ci peuvent intervenir à différents points de l'algorithme, selon la tactique employée :

- soit ils aident à réduire l'espace des solutions à visiter en évitant les solutions qui ne sont pas suffisamment significatives. Nous considèrerons que ces algorithmes sont des heuristiques ;
- soit ils interviennent après l'extraction faite uniquement sur des critères structurels, afin de classifier les résultats obtenus. Dans ce cas, nous considèrerons l'algorithme comme exact.

3.4.2.1 Heuristiques

Trois des approches relevées sont basées sur la recherche de k-tuples et cherchent à considérer simultanément plusieurs sites associés dans la composition d'un promoteur. L'espace combinatoire à visiter étant plus vaste dans ce cas, elles utilisent des heuristiques. La méthode présentée dans [Klingenhoff et al., 1999], nommée FastM, est une variation de celle utilisée dans l'outil de comparaison de séquences Fasta ([Pearson and Lipman, 1988], [Pearson, 1990]). Elle cherche à détecter les sites apparaissant dans un ordre conservé, et séparés par un *spacer*.

La méthode décrite dans [Fraenkel et al., 1995] énumère tous les k-tuples de taille réduite (moins de dix symboles) présents exactement ou avec un certain nombre de substitutions dans chacune des séquences. Ces k-tuples peuvent contenir un *spacer* de longueur variable mais réduite. Les k-tuples recueillant les meilleures valeurs de signification sont ensuite utilisés pour construire des motifs plus larges.

L'algorithme de Wolferstetter recherche les k-tuples qui apparaissent dans au moins un certain pourcentage d'un (petit) jeu de séquences, avec une ou plusieurs substitutions. Ces dernières pouvant se trouver à différentes positions dans chacune des occurrences d'un k-tuple, il fait l'hypothèse que les vrais sites à localiser doivent avoir des positions préférentielles pour ces substitutions. La sélection est ensuite affinée par une maximisation des critères de signification ([Wolferstetter et al., 1996]).

Parmi les approches plus récentes, celle décrite dans [Pavesi et al., 2001] est une heuristique inspirée des méthodes exactes que nous utilisons ([Sagot, 1998] et [Marsan and Sagot, 2001], qui seront détaillées dans les chapitres suivants). Elle réduit l'espace des solutions en ne considérant que les occurrences d'un motif consensus qui ont des substitutions réparties de façon homogène, et ce dans une partie des séquences. Puis, elle recherche les motifs sans contrainte de répartition, cette fois à partir des motifs obtenus. Ces opérations de « filtrage » peuvent être répétées plusieurs fois afin d'augmenter les chances de ne pas manquer un motif. Des critères de signification sont ensuite appliqués aux motifs pour les classer.

D'autres approches heuristiques récentes font des restrictions similaires sur les sites à détecter (pas de substitutions ou contraintes de répartition sur celles-ci) pour limiter l'espace de recherche : [Brazma et al., 1998b], [Li et al., 1999], [Califano, 2000] et [Gelfand et al., 2000]. L'algorithme développé par [Apostolico et al., 2000] utilise simultanément des critères structurels et de signification pour inférer des mots exacts sur- ou sous-représentés.

Une méthode originale a été récemment présentée dans [Buhler and Tompa, 2001]. Elle consiste, pour détecter un motif consensus de longueur l avec e substitutions, à effectuer des projections depuis l'espace à l dimensions de ce type de motif vers un sous-espace à $k < l - e$ dimensions choisi aléatoirement. L'algorithme sélectionne pour cela une projection, ce qui revient à choisir k positions aléatoirement. Puis, il « projette » (cela consiste à concaténer les symboles se trouvant à ces k positions) toutes les occurrences des séquences. En répétant ces opérations pour différentes projections choisies aléatoirement, on espère voir un certain nombre d'occurrences se projeter sur un même mot de longueur k, ce qui signifierait que les k positions choisies sont les positions fortes d'un consensus de longueur l. Cette approche suppose donc que les erreurs sont réparties sur certaines positions seulement des occurrences d'un motif. Les motifs produits lors de la phase de projection sont ensuite affinés par un algorithme de type EM. Cette méthode semble être assez efficace dans la détection de motifs très dégénérés. La détection de ce type de motifs a été posée comme un « *challenge* » en matière d'inférence de sites de fixation dans [Pevzner and Sze, 2000] (ce dont nous discuterons au terme de ce chapitre). Une des deux méthodes présentées dans cet article est une heuristique qui consiste à sélectionner chaque occurrence possible, à trouver l'occurrence qui en est la plus proche dans chacune des séquences, et à opérer à partir de ce jeu d'occurrences une optimisation à l'aide d'une fonction de score, visant à converger vers le meilleur motif possible.

Enfin, l'approche décrite dans [Eskin and Pevzner, 2001] cherche à détecter des motifs dits composites, composés de deux sites séparés par un *spacer*. Pour cela, elle extrait les sites respectant certains critères structurels selon deux méthodes : celle décrite dans [Sagot, 1998] (exacte) ou bien celle que nous venons de présenter. Elle sélectionne ensuite les motifs qui obtiennent les meilleurs scores de signification et essaie de les associer en paires.

3.4.2.2 Exactes

Les premières approches exactes d'inférence appliquée à des séquences biologiques utilisaient des motifs consensus simples et généraient l'ensemble des motifs possibles ([Queen et al., 1982], [Waterman, 1984], [Galas et al., 1985], [Waterman, 1986], [Staden, 1989], [Waterman, 1989]). Les méthodes développées par Queen *et al.* et Waterman imposent la présence d'au moins une occurrence du motif dans chaque séquence du jeu, tandis que celle de Staden *et al.* permet une couverture incomplète. Le principal défaut de ces approches est que la longueur des motifs est limitée, puisque le nombre de motifs à générer croit exponentiellement avec cette longueur et avec le nombre de substitutions autorisées. Directement issu de ce type d'approches énumératives, Van Helden *et al.* ont développé un algorithme similaire ([van Helden et al., 1998]) à la différence près que plusieurs méthodes d'évaluation de signification sont appliquées aux motifs obtenus. Il souffre des mêmes faiblesses que les méthodes dont il s'inspire puisque les motifs ont une longueur maximale très limitée (6 dans l'article, 8 sur le programme en ligne). De plus, il traite des motifs sans erreurs (ce qui permet en contrepartie de ne générer que les motifs apparaissant dans le texte). Van Helden a étendu cette méthode à la détection de paires de motifs séparés par un *spacer* de taille fixe ([van Helden et al., 2000]). La complexité de l'espace des solutions à considérer (et à générer) entraîne une restriction drastique de la longueur des motifs (3 ou 4 symboles chacun) et de la longueur du *spacer* (20 symboles au maximum).

De nombreuses méthodes, ayant recours à des propriétés des motifs consensus, et éventuellement à des structures de données complexes (arbres des suffixes, tableaux de suffixes, automates), évitent de générer l'ensemble des motifs possibles pour une longueur donnée.

Certaines d'entre elles misent sur la capacité à générer des motifs consensus sur un alphabet dégénéré ([Saurin and Marlière, 1987], [Neuwald and Green, 1994], [Jonassen et al., 1995], [Ulyanov and Stormo, 1995], [Jonassen, 1997], [Brazma et al., 1998b]). Les approches de Neuwald *et al.* et Jonassen *et al.* sont développées pour extraire des motifs diagnostics dans des séquences protéiques mais peuvent être adaptées au traitement de séquences nucléiques. Neuwald utilise un alphabet dégénéré restreint (un symbole ne peut représenter plus de deux monomères) et se sert de critères de signification pour éviter de générer certains motifs trop peu significatifs. L'extraction de motifs réalisée n'est donc pas exhaustive. L'algorithme de Jonassen accepte quant à lui un alphabet dégénéré sans contraintes ainsi que des *spacers*. Ces deux approches sont les seules, celle de Staden exceptée et parmi les approches combinatoires exactes décrites, à considérer des motifs n'ayant pas d'occurrences dans l'intégralité des séquences. Elles ne traitent cependant que des motifs exacts.

Plus récemment, la méthode développée dans [Tompa, 1999] résoud le problème d'inférence en autorisant une couverture partielle des séquences et permet une substitution, mais la combinatoire développée limite la longueur des motifs a des petites valeurs. Dans [Sinha and Tompa, 2000], ce type d'approche énumérative est étendue aux motifs acceptant les *spacers* et construits sur un alphabet dégénéré. Elle se limite de la même manière aux motifs de longueur réduite (6 à 10 bases en tout, hors *spacer*) et les position et longueur du *spacer* sont aussi très contraintes.

Les algorithmes développés par Sagot *et al.* offrent plus de souplesse sur des motifs consensus simples ([Sagot et al., 1995a], [Sagot and Viari, 1996], [Sagot et al., 1997], [Sagot, 1998]), traitant le problème de manière exacte sans générer non plus l'intégralité des motifs possibles. Ces motifs acceptent les substitutions, les *gaps* ou des distances d'édition plus complexes entre motifs. Leurs occurrences ne recouvrent pas nécessairement l'intégralité des séquences, ils peuvent être exprimés sur des alphabets dégénérés incluant les jokers, et peuvent être étendus à la représentation de motifs structuraux dans des séquences d'ARN ou de protéines ([Sagot et al., 1997], [Sagot and A.Viari, 1997]). Comme dans les approches de Neuwald ou Jonassen, les motifs de longueur k sont étendus à partir de ceux de longueur $k-1$. Seuls les motifs valides sont générés. Cette approche est suffisamment efficace pour permettre le traitement de gros jeux de données. C'est sur cette approche que nous avons fondés nos travaux, nous la décrirons plus en détails dans le chapitre qui suit.

Dans [Pevzner and Sze, 2000], une méthode exacte est présentée qui consiste, pour détecter un motif de longueur l avec e substitutions, à construire dans un espace à l dimensions un graphe dont chaque noeud est une occurrence du jeu de séquences, et dont les arcs relient les noeuds séparés par une distance de Hamming inférieure à $2e$. L'objectif est ensuite d'isoler les cliques de ce graphe, les occurrences d'une telle clique formant en fait le e-voisinage d'un motif consensus. Pour ce faire, leur algorithme élimine progressivement des arcs en filtrant ceux qui ne peuvent appartenir à des cliques. Les complexités annoncées semblent rendre le traitement de gros jeux de données difficile.

3.4.3 Discussion

L'état de l'art en matière d'inférence de sites de fixation laisse apparaître quelques grandes orientations pour les algorithmes actuels. La majorité d'entre eux (et les plus utilisés) sont en effet des méthodes d'optimisation. Leur choix est de privilégier l'optimisation des critères de signification pour guider l'inférence. La complexité du problème d'inférence posé en ces termes nécessite l'utilisation d'algorithmes heuristiques pour le résoudre.

Les outils existants utilisent en majorité la représentation matricielle, les motifs consensus étant considérés comme moins aptes à décrire les spécificités des sites de fixation. Cependant, les matrices pondérées classiques (ou bien les HMM, très proches) ne font pas état de résultats probants dans ce domaine, et les approches les utilisant s'avèrent souvent être très sensibles au bruit. Il semble donc qu'il manque dans les représentations existantes certains critères primordiaux qui permettraient de modéliser des sites de fixation avec sensibilité. L'estimation de la signification d'une représentation matricielle, en général de type *log-likelihood ratio*, pourrait de plus ne pas être adéquate pour quantifier la valeur d'un site, l'additivité des entropies de chaque position étant discutable. En effet, pourquoi les sites de fixation, macro-molécules complexes, s'affranchiraient-ils d'une propriété omniprésente en physique moléculaire (et dans bien d'autres domaines), à savoir que les propriétés d'un objet ne peuvent en général se déduire de celles de ses éléments ? Les sites de fixation doivent par conséquent, de notre point de vue, être considérés dans leur globalité, et non comme de simples associations de nucléotides ayant certaines particularités.

Cet état de l'art laisse donc entrevoir une tendance visant à utiliser des représentations plus spécifiques aux sites à inférer. Cela implique une augmentation du rapport signal/bruit, puisqu'un signal complexe a moins de chance d'apparaître par hasard qu'un motif simple. Toute la difficulté du problème réside dans le compromis à faire entre la complexification nécessaire de la représentation utilisée, et celle du problème algorithmique qu'elle implique. Visant à étendre la représentation utilisée, certains algorithmes permettent que les couvertures des sites ne soient pas totales, que les sites soient modélisés sur un alphabet dégénéré, prennent en compte des corrélations entre positions d'un site ou détectent les associations de deux sites séparés par un *spacer*. Certaines de ces caractéristiques sont motivées par des observations faites sur des sites déterminés expérimentalement, et elles semblent être parfois biologiquement nécessaires à la description correcte de ces sites.

Parmi celles-ci, nous avons vu au début de ce mémoire que l'association de plusieurs sites est explicable par la conformation spatiale de certains facteurs qui viennent s'y fixer et par d'éventuelles interactions entre plusieurs d'entre eux (voir l'exemple de la figure 1.3 page 27). Les observations font cependant état dans certains cas d'une distance variable qui sépare deux sites associés (due à une certaine souplesse du complexe enzymatique), et d'associations de plus de deux sites. Les algorithmes d'optimisation, ayant déjà beaucoup de paramètres structurels à estimer, peinent à traiter de telles caractéristiques supplémentaires, et les combinatoires privilégient les méthodes heuristiques pour résoudre le problème de l'inférence de deux sites associés. À ce jour, aucune approche exacte n'existe qui permette l'extraction de deux sites ou plus, associés par des *spacers* de longueur variable. Les seules approches inférant des associations de deux sites ne considèrent que des motifs sans substitutions (éventuellement sur un alphabet dégénéré) ou ne sont pas exactes.

Si, récemment, quelques approches ont été consacrées à l'inférence de motifs aux propriétés étendues pour mieux détecter des associations de sites de fixation, elles souffrent à notre avis de plusieurs lacunes :

- dans [Pevzner and Sze, 2000] est défini un *challenge* en matière d'inférence de sites de fixation, qui consiste à extraire des motifs très dégénérés. L'augmentation du nombre de substitutions possibles est présentée comme un moyen de rendre les motifs plus sensibles à la détection de sites de fixation. La seule application biologique présentée dans cet article montre un site dans lequel les substitutions sont utilisées de manière contigüe, au centre du motif, formant donc un *spacer*. Cet exemple vient appuyer notre opinion, qui est que, en matière de sites de fixation, l'usage d'un *spacer* semble plus pertinent que l'augmentation du nombre de substitutions. Il est, de plus, impossible de traiter de la même manière deux motifs qui seraient séparés par une vingtaine de nucléotides (qui seraient perçus comme autant de substitutions) comme on peut l'observer fréquemment. Le *challenge* présenté dans cet article nous semble donc plus algorithmique que biologique, cherchant à extraire les signaux les plus faibles possibles. Il ne correspond pas aux conditions rencontrées dans le cadre d'une extraction de sites de fixation, surtout que les contraintes imposées dans ce *challenge* impliquent l'utilisation de jeux de données de taille très réduite. Nous soumettons néanmoins nos algorithmes à ce *challenge* dans le chapitre consacré aux tests du programme que nous avons développé afin d'évaluer leurs performances ;
- Eskin présente un algorithme d'inférence de sites constitués de deux motifs séparés par un *spacer* ([Eskin and Pevzner, 2001]). Dans cet algorithme, les motifs sont d'abord extraits individuellement, puis les meilleurs d'entre eux (selon un critère de signification) sont posés comme candidats à une éventuelle association. Cette approche reflète une conception qui, à notre avis, fait l'erreur de considérer un site de fixation contenant un *spacer* comme une simple association de deux motifs significatifs. Rien ne prouve (et nous verrons des contre-exemples dans nos applications) qu'un site de fixation de ce type, sensé être exceptionnellement représenté, soit composé de deux

motifs qui sont également exceptionnellement représentés ;
- au cours de l'année 2002, les mêmes auteurs publieront un article d'applications ([Sze et al., 2002]). Y sont testés différents outils d'inférence (dont le leur) sur des données biologiques dont les sites ont été isolés ou sont inconnus. Parmi les cinq sites inférés, au moins deux sites contiennent un *spacer* qui est perçu comme une série de substitutions par tous les outils utilisés, ce qui rend ces sites considérablement plus faibles. De plus, un des sites a des occurrences contenant des *spacers* de longueurs différentes, ce qu'aucun outil ne gère non plus. Cette flexibilité affaiblit encore le signal si elle n'est pas prise en compte. Ces expériences font au premier abord état d'inférences de motifs très longs (jusqu'à 29 bases) et très dégénérés (41% de substitutions), mais, en considérant le *spacer* que contiennent certains d'entre eux, on se retrouve face à l'inférence de paires de motifs plus courts et mieux conservés. Il y a ici à notre avis une réelle lacune de représentation qui rend le problème plus complexe qu'il ne l'est.

Aucune de ces approches ne gère le fait que le *spacer* utilisé puisse être de longueur variable dans les différentes occurrences d'un même site.

Les connaissances des interactions ADN/protéines se sont considérablement enrichies durant les vingt dernières années (voir l'article de revue [Garvie and Wolberger, 2001]), et les critères structurels de ces interactions chez certains organismes sont mieux connus. Les génomes disponibles sont aussi de plus en plus nombreux. Il convient de se poser plusieurs questions importantes prenant en compte ces changements avant de développer des algorithmes d'inférence pour ces sites :

- quelles caractéristiques structurelles prendre en compte afin d'augmenter le rapport signal/bruit des sites de fixation ? Ces caractéristiques seront-elles à déterminer lors de l'inférence ou seront-elles connues *a priori* ?
- les critères de signification sont-ils fiables au point d'être ceux qui guident l'inférence ?
- quelles quantités de données génomiques sommes-nous appelés à traiter ? Peut-on compter sur une bonne couverture des sites dans les séquences fournies ?

Les réponses à ces questions nous fournissent des orientations à suivre dans nos travaux. Concernant le premier point, les propriétés exposées au terme du chapitre 1 apportent quelques éléments de réponse. Il semble que la prise en compte de sites multiples associés entre eux soit d'importance chez nombre d'organismes, comme le montrent certains sites déterminés expérimentalement et la tendance à prendre en compte cette caractéristique dans les algorithmes récents. Une certaine flexibilité des *spacers* semble également d'intérêt. Il est à noter que les *spacers* sont susceptibles d'avoir des longueurs très importantes, certains sites d'activation se trouvant loin en amont du début de la transcription et pouvant interagir (directement ou indirectement) avec le complexe enzymatique qui initie cette transcription. Un algorithme d'inférence pertinent doit donc être capable de gérer ces associations souples entre différents sites, et de modéliser des *spacers* potentiellement très longs (ce qui exclut de les traiter comme une suite de substitutions). Parallèlement, et toujours dans le but d'affiner la représentation des sites, la prise en compte de corrélations entre positions d'un site pourrait également être d'importance. Si nos travaux se sont moins focalisés sur la modélisation de cette caractéristique, nous verrons que les algorithmes et outils que nous avons développés gèrent de façon limitée certains types de corrélations entre positions.

Concernant le second point, nous pensons que les critères de signification ne devraient pas guider à eux seuls l'inférence de sites de fixation. La complexité croissante des caractéristiques des sites à inférer, nécessaire pour accroître les capacités de détection, et la faible couverture de ces sites rendent les approches d'optimisation très complexes, puisqu'un nombre important de critères structurels doit être estimé. Les algorithmes fondés sur l'optimisation de ces critères de signification approchent le problème sous un angle particulièrement complexe. De plus, ces critères restent arbitraires, et on ne sait à ce jour s'ils sont véritablement pertinents ou non pour exprimer la valeur fonctionnelle d'un site. Notre orientation est donc de réaliser l'inférence de sites sur des critères structurels (étape que nous nommerons *extraction*), et d'effectuer une évaluation statistique *a posteriori*, selon diverses méthodes, pour classifier les résultats obtenus. Cette approche impose à l'utilisateur de fournir certaines caractéristiques structurelles pour inférer des sites.

Sur le dernier point, la profusion actuelle de données génomiques, la difficulté de la localisation expérimentale des sites de fixation dans ces données, et le peu d'assurance qu'apportent les données d'expression fournies par les puces à ADN concernant la couverture d'un site dans des séquences co-exprimées nous amènent à penser qu'un algorithme d'inférence doit être capable de traiter de grands jeux de données dans lesquels un site peut avoir une très faible couverture.

Forts de ces considérations, nos travaux sont donc résolument orientés vers la conception d'algorithmes combinatoires exacts pouvant prendre en compte un grand nombre de critères structurels variés. Nos premiers travaux consisterons à étendre les motifs consensus à une représentation plus complexe, augmentant ainsi leur spécificité jugée trop faible. Nous développerons ensuite des algorithmes adaptés à cette représentation.

Notre objectif est de développer un outil pertinent, efficace et sensible, pour des biologistes ou bio-informaticiens cherchant à inférer des sites de fixation dont ils connaissent certaines caractéristiques. À notre avis, la complexité et la diversité constatées dans les associations de sites de fixation et dans les conditions d'inférence font qu'un algorithme peut difficilement évaluer à lui seul tous les paramètres possibles. L'outil d'inférence que nous avons développé fonctionne en interaction avec l'utilisateur, qui doit fournir des pré-requis et déduire des résultats obtenus les orientations à prendre pour de nouvelles inférences.

Le chapitre qui suit présente l'algorithme de Marie-France Sagot sur lequel nos travaux se fondent.

Chapitre 4

Extraction de modèles simples

Nous traitons dans ce chapitre de l'algorithme développé par Marie-France Sagot dans [Sagot, 1998] et sur lequel sont basés nos travaux.

Cet algorithme est de type combinatoire exact et manipule des motifs consensus. Il permet de réaliser l'extraction de motifs consensus respectant certains critères structurels simples d'un jeu de séquences. Il n'utilise pas de critères de signification et n'a donc pas pour objectif en l'état d'inférer directement des sites de fixation. Les motifs utilisés peuvent contenir des substitutions et des *gaps*.

Une notion introduite dans [Sagot et al., 1995a] y est utilisée : le *modèle*, qui est un motif consensus soumis à un critère de quorum. Les séquences traitées sont stockées dans une structure d'arbre des suffixes. Ces choix sont décrits et motivés dans la section 4.1.

L'algorithme d'extraction de modèles répétés dans une séquence est ensuite présenté dans la section 4.2. Cette instance restreinte du problème d'extraction de modèles communs à plusieurs séquences nous permet d'introduire l'algorithme de façon simplifiée. Ceci fait, nous étendons le problème à celui des modèles communs à plusieurs séquences dans la section 4.3.

Enfin, nous discutons du positionnement de cet algorithme par rapport au problème biologique des sites de fixation que nous souhaitons traiter, de ses limitations, et des extensions à y apporter. Les travaux qui en ont découlés et qui font l'objet de cette thèse seront décrits dans le chapitre qui suit.

4.1 Choix des représentations et structures

4.1.1 Modèles

Afin de lever l'ambiguïté due à l'usage du terme « motif » pour décrire indifféremment des mots exacts ou approchés, une occurrence dans la séquence ou un consensus extérieur à la séquence, Marie-France Sagot utilise la notion de modèle, proche de celle de motif consensus.

Un modèle est un motif consensus extérieur à la séquence, auquel sont intégrés deux critères :
- le nombre maximum de substitutions permis ;
- le quorum, indiquant le nombre minimum d'occurrences qu'il doit regrouper.

Précisons que les algorithmes présentés dans ce mémoire ne gèrent que les substitutions. L'algorithme original décrit également des modèles autorisant les *gaps*. Notre description se limitera à l'usage des substitutions pour deux raisons :
- algorithmique : le problème est évidemment beaucoup moins coûteux dans ce cas, l'espace des solutions étant en général nettement moins vaste ;
- biologique : dans le domaine des signaux que nous étudions, on observe généralement moins de *gaps* que de substitutions. Le gain en sensibilité qu'ils apportent pourrait ne pas être très important.

L'extension de tous les algorithmes présentés dans ce mémoire à la gestion des *gaps* ne représente cependant pas une difficulté importante, mais les optimisations propres à ce type de distance d'édition n'ont pas été poussées plus avant pour le moment.

Un modèle est défini de la façon suivante :

Définition 4 (Modèle et occurrences d'un modèle)
On appelle modèle $m \in \Sigma^+$ *la représentation de l'ensemble* $\mathcal{M}$ *des mots d'une séquence, défini par :*

$$\mathcal{M} = \{x \in \Sigma^*, \mathcal{D}_H(m, x) \leq e\}$$

où $\mathcal{D}_H$ *représente la distance de Hamming (définie à la section 2.3.2.1 page 43), et* e *un entier représentant le nombre maximal de substitutions associé au modèle.*

On appelle occurrence du modèle m *tout mot de la séquence qui appartient à cet ensemble* $\mathcal{M}$*.*

Pour le moment identique aux motifs consensus, un modèle est un objet extérieur à la séquence : le mot qui le définit n'apparaît pas nécessairement dans la séquence. La figure 4.1 montre l'exemple d'un modèle et de ses occurrences dans une séquence donnée.

Définition 5 (Modèle valide)
Un modèle est dit valide *s'il vérifie la contrainte de quorum qui lui est imposée, c'est-à-dire si le nombre de ses occurrences est supérieur ou égal au quorum* q *associé au modèle.*

Dans l'exemple de la figure 4.1, le modèle $CGCA$ a six occurrences dans la séquence. Si on lui attribue par exemple un quorum de quatre occurrences, ce modèle est considéré comme valide. Le modèle $AAAA$ (avec une substitution et le même quorum) ne possède que deux occurrences et ne sera donc pas considéré comme valide.

D'un point de vue algorithmique, un modèle de longueur k avec e substitutions est, comme un motif consensus, une boule de rayon e dans un espace à k dimensions. Les modèles ne sont pas des cliques [1] : pour une clique constituée d'occurrences dont chaque

[1] une clique est un ensemble dont tous les éléments respectent deux à deux une propriété donnée.

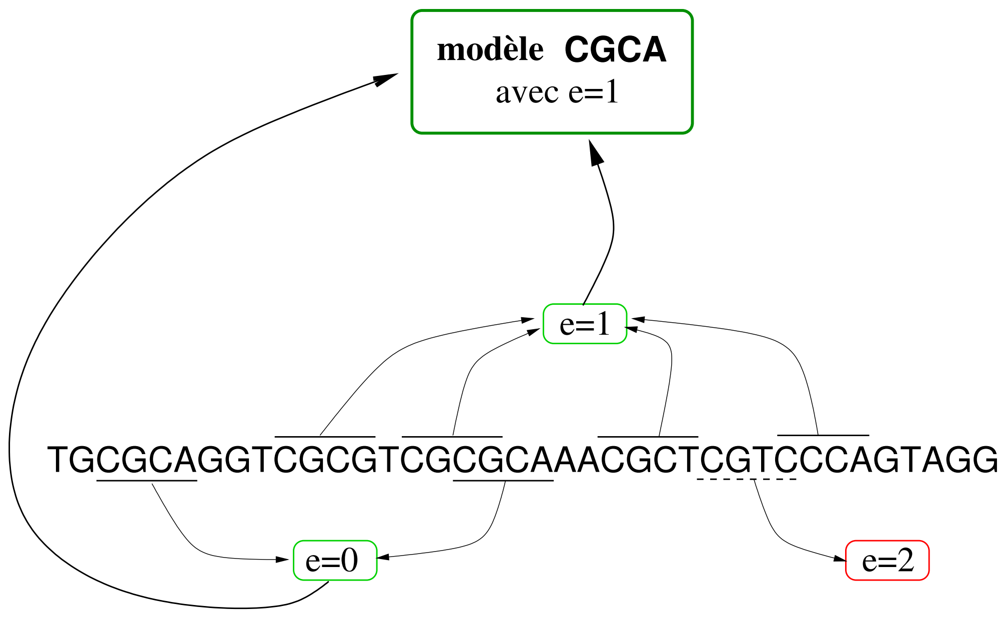

FIG. 4.1 – Un modèle et ses occurrences.
On montre ici le modèle $CGCA$ avec 1 substitution, et ses occurrences dans la séquence. Certains mots sont à la bonne distance du modèle et sont donc des occurrences de ce modèle. D'autres mots sont trop éloignés et ne lui appartiennent donc pas.

couple a la propriété d'appartenir à un modèle à e substitutions, il n'existe pas nécessairement de modèle à e substitutions contenant toutes les occurrences de cette clique. Par exemple, l'ensemble $\{ACAG, AGAC, AAAA\}$ est une clique car tout couple formé à partir de ses éléments respecte la propriété d'appartenir à un modèle acceptant une substitution (parmi $ACAC, AGAG, AGAA, ACAA, AAAG$ et $AAAC$). Pourtant, aucun de ces modèles ne contient les trois occurrences de l'ensemble initial. Projeté dans le plan, il est par exemple évident qu'il n'existe pas de point du plan tel que les trois sommets d'un triangle équilatéral de côté $2r$ (et respectant donc tous une propriété de distance) se trouvent à l'intérieur d'un cercle centré en ce point et de rayon r.

Les modèles ne sont pas non plus des chaînes de Steiner [2]. Par exemple, l'ensemble $S = \{AAAA, AAAA, ATAT\}$ a pour unique chaîne de Steiner $AAAA$. Or, avec une substitution permise, $AAAA$ n'est pas un modèle ayant pour occurrences les mots de S. Seuls $ATAA$ et $AATA$ sont de tels modèles, et ce ne sont pas des chaînes de Steiner de S. Un modèle est en effet le « centre » d'une boule dans laquelle doivent être contenues ses occurrences, alors qu'une chaîne de Steiner est un « barycentre » d'un ensemble d'occurrences.

[2] pour une distance $\mathcal{D}$ et un ensemble de mots S, un mot m est une chaîne de Steiner de S si, pour tous les mots $u \in S$, $\Sigma_{u\in S}\mathcal{D}(m, u)$ est minimal.

La notion de similarité utilisée dans les modèles peut être considérée de façon plus large que dans la définition précédente ([Sagot and Wakabayashi, 2001]). On peut par exemple concevoir des relations non-transitives entre symboles de l'alphabet et exprimer les modèles sur un alphabet dégénéré.

4.1.2 L'arbre des suffixes

Les algorithmes d'extraction décrits dans ce chapitre utilisent une structure d'arbre des suffixes pour stocker la ou les séquences à traiter. Nous nous contenterons de donner dans cette partie les propriétés de base de cette structure de données, et d'expliquer sa construction. On trouvera de plus amples détails sur ces structures et leur construction dans [McCreight, 1976] et [Ukkonen, 1995], et une description plus générale des structures utilisées en algorithmique du texte dans [Crochemore et al., 2001] et [Gusfield, 1997].

4.1.2.1 Définition

La figure 4.2 offre tout d'abord un résumé des nomenclatures que nous utiliserons pour traiter des arbres.

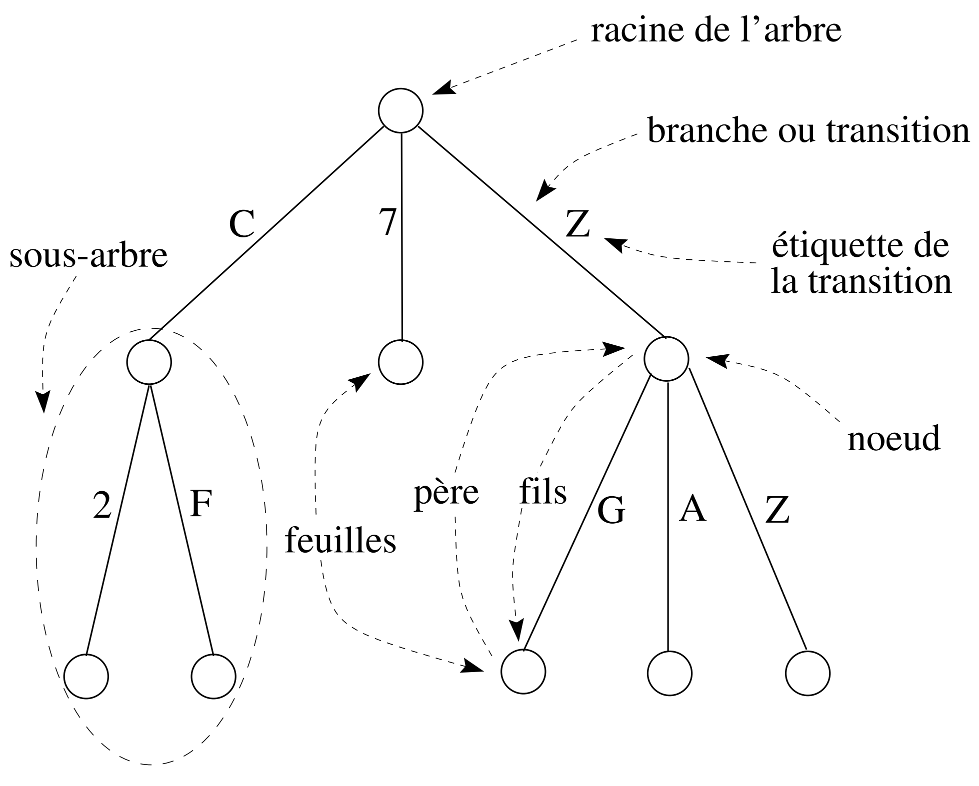

FIG. 4.2 – Un arbre et les nomenclatures associées.

Un arbre des suffixes est une structure de données permettant de stocker une séquence et d'accéder rapidement à un de ses facteurs. Chaque chemin de cet arbre allant de la racine à une feuille épelle un suffixe de la séquence à partir de laquelle il a été construit. Tous les suffixes, de la séquence complète au mot vide, sont représentés. Chaque arc est étiqueté par un symbole de la séquence. À chaque facteur de la séquence correspond, lorsqu'on le lit depuis la racine, un noeud de l'arbre : on parle de *noeud-occurrence*.

Définition 6 (Noeud-occurrence)
Dans un arbre des suffixes $\mathcal{A}$ construit à partir de la séquence s, un noeud-occurrence v représente en même temps un noeud de l'arbre et chaque occurrence dans s du mot épelé depuis la racine jusqu'à ce noeud. On parlera donc indifféremment de v comme d'un noeud de l'arbre ou comme d'une occurrence.

Afin que la présence d'un suffixe étant également un facteur dans la séquence soit signifiée, on ajoute un symbole ne figurant pas dans l'alphabet (dans notre cas '\$') à la fin de la séquence qui sert à construire l'arbre. La figure 4.3 représente un tel arbre.

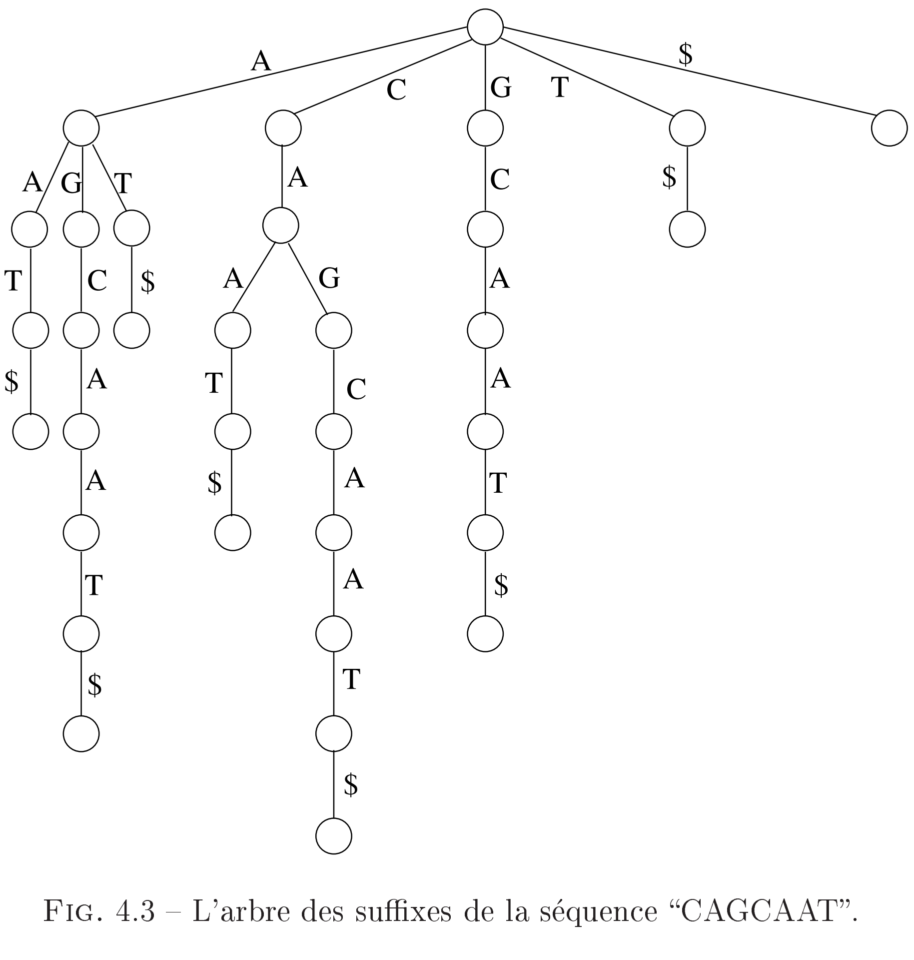

FIG. 4.3 – L'arbre des suffixes de la séquence "CAGCAAT".

Un arbre des suffixes peut être quadratique en espace. L'algorithme d'extraction utilise donc la forme compactée de l'arbre des suffixes, qui possède les propriétés suivantes :
- tout arc de l'arbre contient un facteur non-vide (au lieu d'un symbole) de la séquence ;
- tout noeud de l'arbre qui n'est pas une feuille (exceptée la racine) a au moins deux arcs sortants.

Ainsi, les arcs ne font plus référence à des caractères mais à des chaînes de caractères lorsque cela est possible. Ces chaînes de caractères ne sont pas stockées telles quelles dans les arcs, mais sous la forme (occupant un espace constant) de deux entiers indiquant la position de début et la longueur du facteur concerné dans la séquence.

Bien que cette forme compactée soit utilisée dans tous les algorithmes décrits dans ce rapport, nous considérerons, dans un souci de clarté, les arbres utilisés comme des arbres non compactés tel celui de la figure 4.3.

Définition 7 (Pseudo-noeud)
Nous appellons pseudo-noeud *de l'arbre des suffixes compacté $\mathcal{A}_c$ d'une séquence s tout noeud de l'arbre des suffixes $\mathcal{A}$ de s qui n'est pas un noeud de $\mathcal{A}_c$. Un pseudo-noeud est donc un noeud virtuel d'un arbre des suffixes compacté, situé sur une transition.*

Notation 4 (Père d'un noeud)
Étant donné un noeud ou un pseudo-noeud v d'un arbre des suffixes compacté $\mathcal{A}_c$, on note père(v) le noeud ou pseudo-noeud père de v dans $\mathcal{A}_c$.

Les arbres des suffixes que nous utilisons sont munis des liens suffixes définis comme suit dans [McCreight, 1976].

Définition 8 (Lien suffixe)
Si on considère tout noeud p de l'arbre correspondant au facteur αw (avec $\alpha \in \Sigma$ et $w \in \Sigma^*$) et le noeud p' correspondant au facteur w, il y a un lien suffixe *de p vers p'.*

Les liens suffixes ne nous sont utiles que lors de la construction de l'arbre (ils sont les éléments clés qui permettent d'obtenir la linéarité en temps à la construction) et peuvent de ce fait être éliminés de la structure finale. La version compactée et munie de liens suffixes de l'arbre de la figure 4.3 est donnée figure 4.4. Ce sont les propriétés de ces liens suffixes qui permettent de construire un arbre des suffixes en temps et en espace linéaires en la longueur de la séquence.

4.1.2.2 Pourquoi un arbre des suffixes ?

La structure d'arbre des suffixes se construit en temps linéaire en la longueur de la séquence et permet de vérifier la présence d'un mot en un temps linéaire en la longueur de ce mot. Elle permet d'obtenir rapidement les positions de ce mot dans la séquence. Pour un alphabet Σ quelconque, ses complexités de construction sont de $\mathcal{O}(n \log |\Sigma|)$ en temps et $\mathcal{O}(n)$ en espace (avec n la longueur de la séquence), ou bien $\mathcal{O}(n)$ et $\mathcal{O}(n|\Sigma|)$, pour les implantations souhaitant conserver la linéarité de la recherche.

L'algorithme d'extraction que nous présentons dans ce chapitre est articulé autour de cette structure d'arbre des suffixes, comme nous le verrons dans la section suivante. En particulier, les modèles sont construits récursivement, ce à quoi se prêtent bien les arbres.

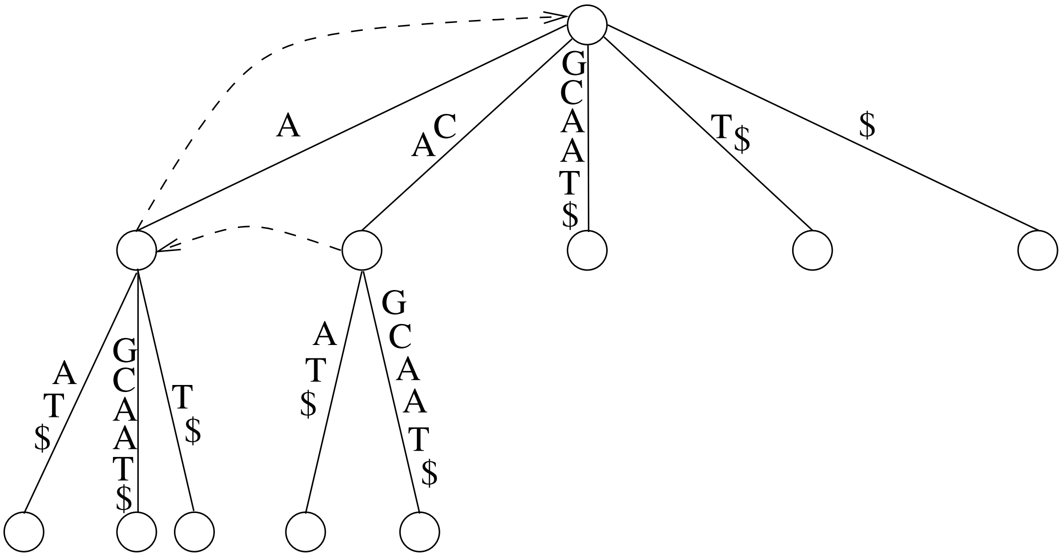

FIG. 4.4 – Un arbre compact des suffixes construit à partir de la séquence "CAGCAAT" (version compactée de l'arbre de la figure précédente).
Les liens suffixes des noeuds internes sont indiqués en pointillés.

L'arbre des suffixes a cependant un inconvénient majeur : il est coûteux en espace bien que linéaire. On parle en général d'une constante de 28 pour l'algorithme de McCreight. Certains algorithmes de construction plus complexes atteignent de meilleures constantes ou une meilleure répartition en mémoire [3] ([Ferragina and Grossi, 1995], [Giegerich et al., 1999], [Kurtz, 1999], [Grossi and Vitter, 2000]). La structure de tableau des suffixes de Myers ([Manber and Myers, 1993]) a une constante en espace de 9, mais elle est mal adaptée à la construction récursive qu'implique l'algorithme.

Quoi qu'il en soit, la structure d'arbre des suffixes utilisée dans les algorithmes de Marie-France Sagot, et dans ceux que nous avons développés, n'est pas indissociable de ces algorithmes d'extraction et peut être remplacée par toute autre structure qui permettrait :

- un parcours récursif efficace, c'est-à-dire, pouvoir passer en temps constant d'une occurrence w à une occurrence $w\alpha$ (avec $w \in \Sigma^+$ et $\alpha \in \Sigma$) et inversement ;
- l'évaluation en temps constant du nombre d'occurrences d'un modèle donné (rendue possible dans l'arbre par de légères modifications lors de sa construction) ;
- la détermination des positions d'un facteur (même remarque).

Nous parlerons dans le chapitre suivant de la structure d'arbre des facteurs que nous avons implantée et qui respecte ces propriétés. L'automate des suffixes (DAWG, voir [Crochemore and Vérin, 1997]), moins coûteux en espace, pourrait également être un bon candidat pour remplacer l'arbre des suffixes dans cet algorithme et ceux du même type.

[3] pour un certain type de parcours, de façon à éviter les fautes de page mémoire durant ce parcours.

4.2 Algorithme d'extraction de modèles répétés dans une séquence

4.2.1 Critères d'extraction

Cet algorithme d'extraction requiert plusieurs critères structurels nécessaires, qui permettent de réduire l'espace des solutions à visiter. Ces critères sont :
- la longueur minimale des modèles ;
- le nombre de substitutions maximal des modèles ;
- le quorum minimal requis pour chaque modèle.

On considérera dans cette section qu'une extraction se fait selon ces trois critères, notés respectivement k_{min}, e, et q.

4.2.2 Principe

Cet algorithme est exact, il permet donc d'obtenir l'intégralité des solutions respectant les critères structurels fournis. Cependant, il ne visite pas l'espace des solutions dans sa totalité, en se basant sur le lemme suivant :

Lemme 2 *([Sagot, 1998])*
Soit un modèle $m \in \Sigma^+$ ayant $occ(m)$ occurrences. Pour tout modèle $m' \in \Sigma^+$ dont m est un préfixe, on a $occ(m') \leq occ(m)$.

Preuve : Étant donné que les occurrences de m' ont une distance d'au plus e avec m', les préfixes de longueur $|m|$ de ces occurrences de m' ont nécessairement une distance d'au plus e avec m, et sont donc toutes des occurrences de m. Le modèle m a donc au moins autant d'occurrences que m'.□

L'algorithme simule le parcours en profondeur d'un arbre virtuel de tous les modèles possibles. Les modèles sont donc étendus par la droite. Simultanément, les occurrences du modèle courant sont également étendues dans l'arbre des suffixes $\mathcal{A}$ de la séquence à traiter, en respectant la règle suivante :

Régle 1
Le couple (v, e_v) est une occurrence du modèle $m' = m\alpha$, avec $m \in \Sigma^*$, $\alpha \in \Sigma$, v un noeud-occurrence de $\mathcal{A}$ et e_v le nombre de substitutions associé, si et seulement si une des conditions suivantes est respectée :
- **(match)** *le couple (père$(v), e_v$) est une occurrence du modèle m et l'étiquette de la transition qui mène de père(v) à v est α ;*
- **(substitution)** *le couple (père$(v), e_v-1$) est une occurrence du modèle m, l'étiquette de la transition qui mène de père(v) à v est $\beta \neq \alpha$, et $e_v \leq e$ (le nombre maximum de substitutions autorisé).*

À chaque occurrence v d'un modèle m est donc associé le nombre de substitutions e_v accumulé, pour former un couple (v, e_v). Dire que ce couple est une occurrence de m signifie que v est une occurrence de m (quel que soit le nombre de substitutions qu'il permette) et que $\mathcal{D}_H(v, m) = e_v$.

Le principe de récursion de l'algorithme est le suivant : soit m un modèle valide (ayant donc $occ(m) \geq q$ occurrences). Pour tout $\alpha \in \Sigma$:

1. On étend chaque occurrence de m d'un symbole vers la droite.
2. Les occurrences qui respectent la distance requise avec $m\alpha$ sont conservées, les autres éliminées.
3. Si $m\alpha$ est valide ($occ(m\alpha) \geq q$) on poursuit son extension de la sorte. Sinon l'extension est interrompue (par le lemme 2, on sait que si un modèle n'obtient pas le quorum requis, tous les modèles qu'on génèrera en l'étendant par la droite ne vérifieront pas non plus ce quorum). Les modèles valides de longueur supérieure à k_{min} sont retenus.

De cette façon, l'espace des solutions (l'ensemble de tous les modèles ayant les critères souhaités) n'est pas visité intégralement. Seuls les modèles dont le quorum indique qu'ils peuvent potentiellement donner naissance à des modèles valides sont étendus. Les figures 4.5 et 4.6 montrent deux étapes successives de l'algorithme.

L'algorithme utilise l'arbre des suffixes décrit dans la section précédente, en y ajoutant certaines informations qui lui sont nécessaires. Le calcul du nombre d'occurrences nécessite d'abord de connaître, pour chaque pseudo-noeud de l'arbre, le nombre de feuilles de son sous-arbre. On stocke donc cette valeur dans chaque noeud de l'arbre des suffixes (ce qui est réalisé en temps et espace linéaires par un parcours en profondeur de l'arbre après construction).

Lors de l'affichage des modèles retenus, si l'on souhaite obtenir les positions des occurrences correspondantes, il faut également stocker dans les feuilles de l'arbre la position dans la séquence du suffixe correspondant (lors de la construction, sans modifier la linéarité en temps et espace).

4.2.3 Conditions d'arrêt

Lors du parcours récursif de l'arbre virtuel des modèles, une descente peut être interrompue pour deux raisons :

- soit le nombre d'occurrences d'un modèle ne respecte plus le quorum requis, comme dans l'exemple de la figure 4.6 ;
- soit la longueur maximum d'un modèle est atteinte, dans le cas où on en spécifie une. Si cette longueur maximum n'est pas spécifiée ($k_{max} = \infty$), seul le critère de quorum met fin à une descente récursive.

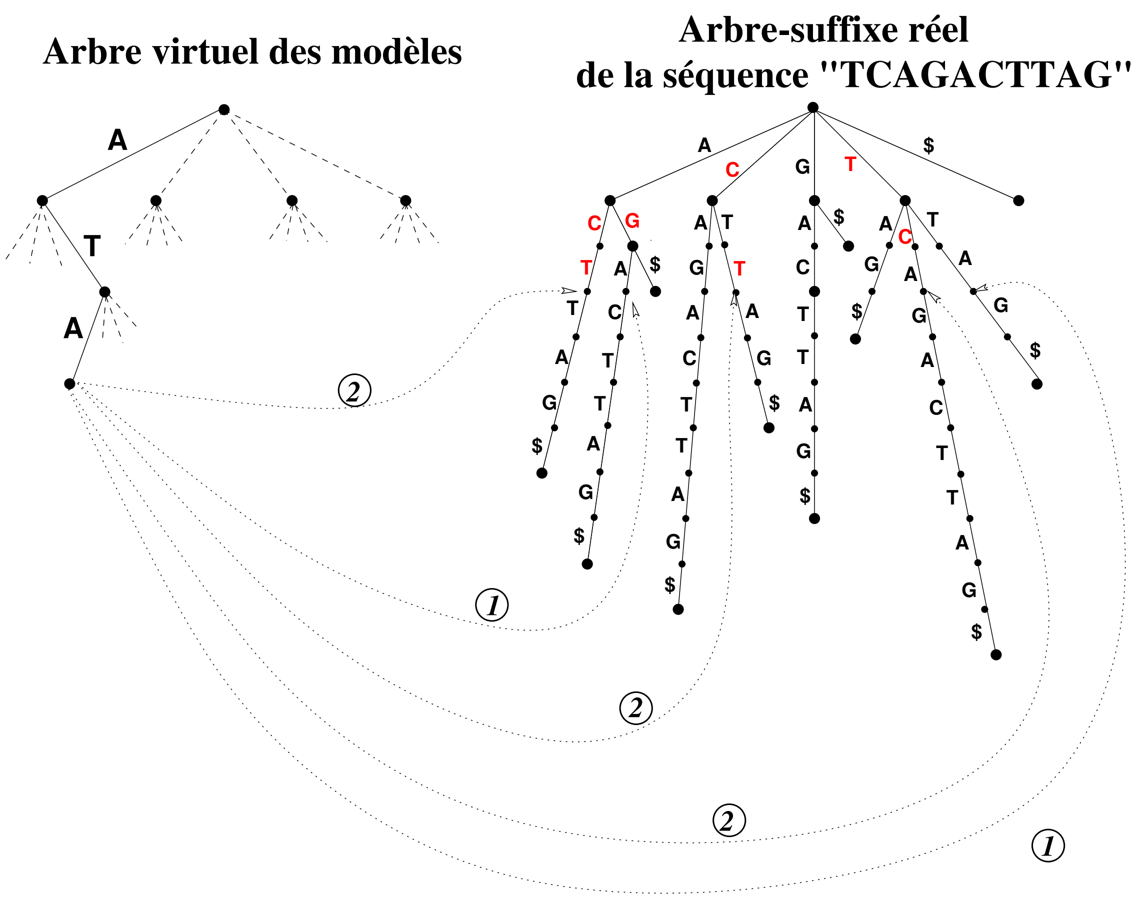

FIG. 4.5 – Une étape de l'algorithme.
Le quorum q est fixé à 4, le nombre maximum de substitutions e à 2. À cette étape le modèle ATA a été exploré et 5 occurrences de ce modèle ont été trouvées dans la séquence. Ce modèle est donc valide. La distance d'édition entre le modèle et chacune de ses occurrences est indiquée sur la flèche qui les relie, chaque substitution effectuée entre le modèle et une occurrence est indiquée en rouge (ou grisé) dans l'arbre. Une fois ce modèle validé, il est étendu avec tous les symboles possibles. La figure 4.6 montre son extension par G.

4.2.4 Pseudo-code

Nous présentons ici le pseudo-code de la fonction d'extension récursive des modèles. Cette fonction prend en paramètres un modèle m valide, sa longueur courante l, et son jeu d'occurrences occ_m. À une occurrence o donnée, on associe toujours la distance o_{err} avec son modèle. Les valeurs k_{min} et k_{max} sont les limites de longueur des modèles (avec éventuellement $k_{max} = \infty$), e est le nombre maximum de substitutions autorisées, q est le quorum requis. La fonction $nbFeuilles_o$ renvoie le nombre de feuilles du sous-arbre dont la

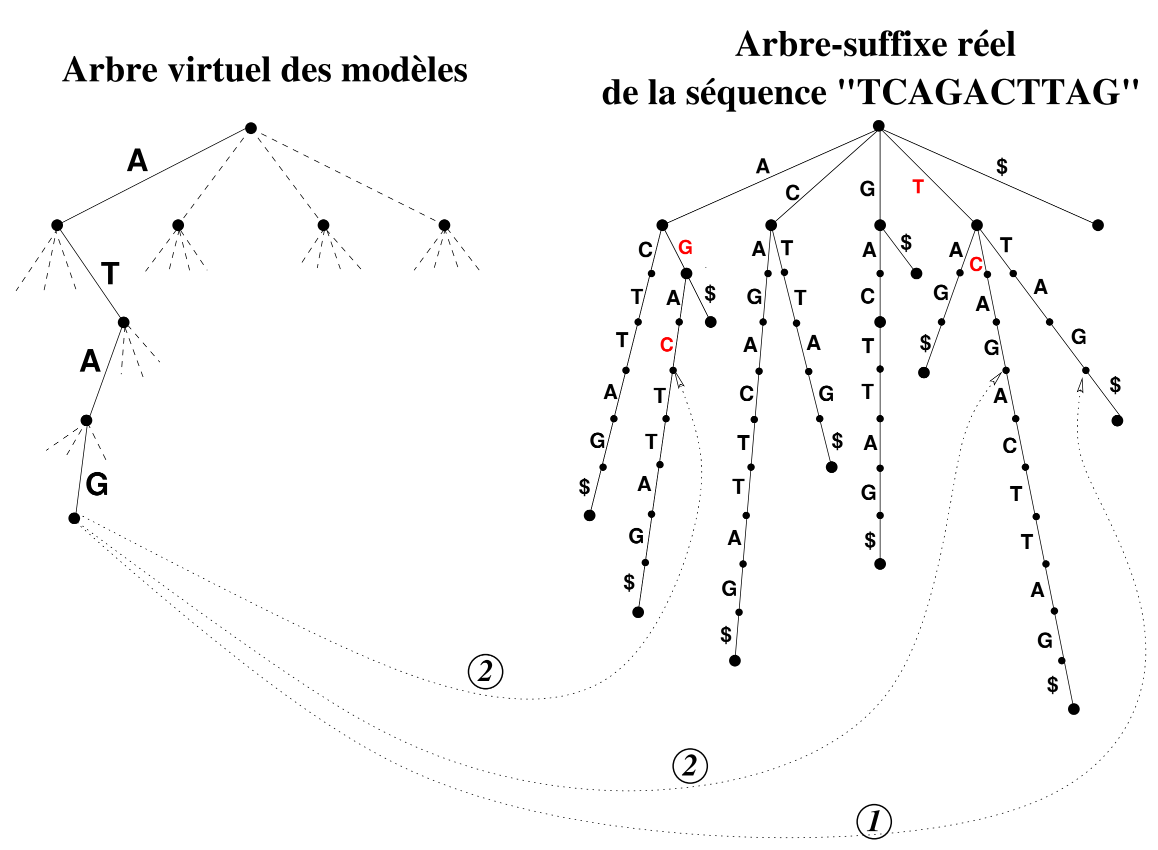

FIG. 4.6 – Suite de l'étape de l'algorithme décrite figure 4.5.
Lors de cette extension de ATA par G, on étend simultanément les occurrences de ATA : deux des cinq occurrences de ATA ne sont plus des occurrences de $ATAG$ car elles ont accumulé trop de substitutions. Restent trois occurrences. Le quorum étant fixé à quatre, on conclut cette étape en invalidant le modèle $ATAG$ (et par là même tous les modèles dont il est le préfixe puisqu'on interrompt la descente récursive dans ce sous-arbre de l'arbre virtuel).

racine est le (pseudo-)noeud correspondant à l'occurrence o. Cette valeur est stockée dans les noeuds de l'arbre des suffixes et s'obtient donc en temps constant pour un noeud donné.

La fonction ÉpelleModèles décrite ici est appelée initialement avec $l = 0$, $m = \varepsilon$ et $occ_m = \varnothing$.

Algorithme A

```
ÉpelleModèles(Longueur $l$, Modèle $m$, Occurrences $occ_m$)
Si $k_{min} \leq l \leq k_{max}$ Alors
  AfficheModèle($m, occ_m$)
FinSi
Si $l < k_{max}$ Alors /* peut-on poursuivre l'extension ? */
  Pour tout symbole $\alpha \in \Sigma$ Faire
    $nb_{occ} = 0$
    $occ_{m\alpha} = \emptyset$
    Pour tout couple $(x, x_{err}) \in occ_m$ Faire
      Si $x$ est un noeud de l'arbre Alors
        Pour tout symbole $\beta \in \Sigma$ tel qu'il existe une transition
          par $\beta$ partant de $x$ vers un (pseudo-)noeud $x'$ Faire
          Si $\alpha \neq \beta$ Alors $x'_{err} = x_{err} + 1$
          Sinon $x'_{err} = x_{err}$
          FinSi
          Si $x'_{err} \leq e$ Alors
            Ajouter le couple $(x', x'_{err})$ à $occ_{m\alpha}$
            $nb_{occ} = nb_{occ} + nbFeuilles_{x'}$
          FinSi
        FinPour
      Sinon /* $x$ est alors un pseudo-noeud */
        Soit $\beta \in \Sigma$ l'unique transition possible par $x$ et menant à $x'$
        Si $\alpha \neq \beta$ Alors $x'_{err} = x_{err} + 1$
        Sinon $x'_{err} = x_{err}$
        FinSi
        Si $x'_{err} \leq e$ Alors
          Ajouter le couple $(x', x'_{err})$ à $occ_{m\alpha}$
          $nb_{occ} = nb_{occ} + nbFeuilles_{x'}$
        FinSi
      FinSi
    FinPour

    Si $nb_{occ} \geq q$ Alors
      ÉpelleModèles($l + 1, m\alpha, occ_{m\alpha}$)
    FinSi
  FinPour
FinSi
```

4.2.5 Complexités

Comme indiqué précédemment, la construction de l'arbre des suffixes d'une séquence s de longueur n sur un alphabet de taille fixée se fait en temps et en espace $\mathcal{O}(n)$.

L'ajout d'une information dans chaque noeud concernant le nombre d'occurrences de chaque facteur de la séquence se fait en temps $\mathcal{O}(n)$ par un parcours en profondeur de l'arbre, et en espace constant pour chaque noeud de l'arbre.

Les complexités de cet algorithme sont les suivantes :

Lemme 3 *([Sagot, 1998])*
Considérant l'extraction sur un alphabet de taille constante de modèles de longueur fixée k avec e substitutions au plus, dans une séquence de longueur n, l'algorithme produit tous les modèles valides en temps $\mathcal{O}(n\mathcal{V}(e,k))$, où $\mathcal{V}(e,k)$ représente le cardinal du voisinage d'un modèle [4].

Preuve : Valider un modèle de longueur k nécessite de lui trouver au moins q occurrences de longueur k en parcourant l'arbre des suffixes jusqu'à la profondeur k. À ce niveau de l'arbre, il existe $p \leq n$ noeuds. Or chacun de ces noeuds peut être visité plusieurs fois, étant donné qu'il fait partie des e-voisinages d'au plus $\mathcal{V}(e,k)$ modèles distincts. La valeur de $\mathcal{V}(e,k)$ est donc une borne supérieure au nombre de visites que peut recevoir chacun de ces p noeuds.

La visite de ces noeuds se faisant en temps constant, le nombre d'opérations à effectuer pour extraire tous les modèles valides d'une séquence dont l'arbre des suffixes a été préalablement construit est majoré par $p\mathcal{V}(e,k)$, qui l'est lui-même par $\mathcal{O}(n\mathcal{V}(e,k))$. □

Lemme 4 *(M.F. Sagot [Sagot, 1998])*
L'extraction de tous les modèles valides de longueur k dans le problème des modèles répétés, sur un alphabet de taille constante, se fait en espace $\mathcal{O}(kn)$, où n est la longueur de la séquence, sans tenir compte de la taille de la sortie produite.

Preuve : L'espace requis pour l'arbre des suffixes que nous utilisons est linéaire. Le stockage des occurrences à étendre pendant le déroulement de l'algorithme requiert un espace supplémentaire : la traversée de l'arbre se faisant récursivement, nous devons conserver les noeuds-occurrences du modèle traité et ceux de tous ses préfixes [5], que nous majorons par kn. La complexité totale en espace est donc en $\mathcal{O}(n + kn) = \mathcal{O}(kn)$.□

[4] le cardinal $\mathcal{V}(e,k)$ du e-voisinage d'un motif de longueur k a été défini dans la définition 2 page 47. Le voisinage d'un modèle ayant les mêmes caractéristiques est identique.

[5] dans le papier original, il est fait état d'un nombre maximum $2n$ pour ces noeuds-occurrences. Nous pensons que cette borne peut être atteinte moyennant un surcoût en temps, et qu'une borne en en doit pouvoir l'être sans surcoût. Toutefois, nous conserverons par précaution une borne kn, en gardant à l'esprit que c'est une borne large et qu'elle peut probablement être améliorée.

4.3 Algorithme d'extraction de modèles communs à plusieurs séquences

L'algorithme dont nous traitons ici est une extension du précédent : nous cherchons maintenant à extraire des modèles communs à plusieurs séquences. Le quorum à atteindre pour valider un modèle concerne dorénavant le nombre de séquences dans lesquelles les occurrences de ce modèle apparaissent (sa couverture en séquences), et non plus le nombre de ses occurrences.

4.3.1 Modifications apportées à la structure de données

L'arbre des suffixes doit dorénavant stocker plusieurs séquences au lieu d'une seule. Ce type d'arbre est dit *généralisé*, et se construit en temps et espace linéairement proportionnels à la somme des longueurs des séquences.

Comme dans le premier algorithme, la validation d'un modèle nécessite de calculer la couverture de son ensemble d'occurrences afin de vérifier si elle respecte le quorum imposé. Cette fois, il nous faut stocker dans chaque noeud p de l'arbre un tableau de bits $Colors_p$ indiquant quelles sont les séquences concernées par les occurrences associées au noeud p. Le tableau de bits d'un noeud est l'union des tableaux de bits de chacun de ses fils.

Pendant le déroulement de l'algorithme, on vérifiera si le modèle en cours respecte le quorum en effectuant l'union des tableaux de bits qui se trouvent dans les noeuds associés à chaque occurrence du modèle. Le nombre de bits à 1 dans le tableau résultant de cette union indique la couverture du modèle, *i.e.* le nombre de séquences dans lesquelles il apparaît au moins une occurrence du modèle.

La figure 4.7 montre l'arbre de deux séquences et les tableaux de bits correspondants.

4.3.2 Modifications apportées à l'algorithme

L'algorithme utilisé pour résoudre le problème des modèles communs est très proche du précédent. Il n'en diffère en fait que par l'utilisation de $Colors$ lors du calcul de couverture d'un modèle.

Notons que $Colors$ s'emploie ici de deux façons :

- $Colors_x$ désigne le tableau de bits associé au pseudo-noeud x ;
- $Colors_m$ désigne le tableau de bits résultat de l'union des tableaux de bits de tous les noeuds-occurrences du modèle m.

Lors de la validation d'un modèle, il nous faut savoir combien de séquences sont concernées par les occurrences trouvées pour ce modèle. Pour ce faire, on calcule l'union des tableaux de bits de ces occurrences.

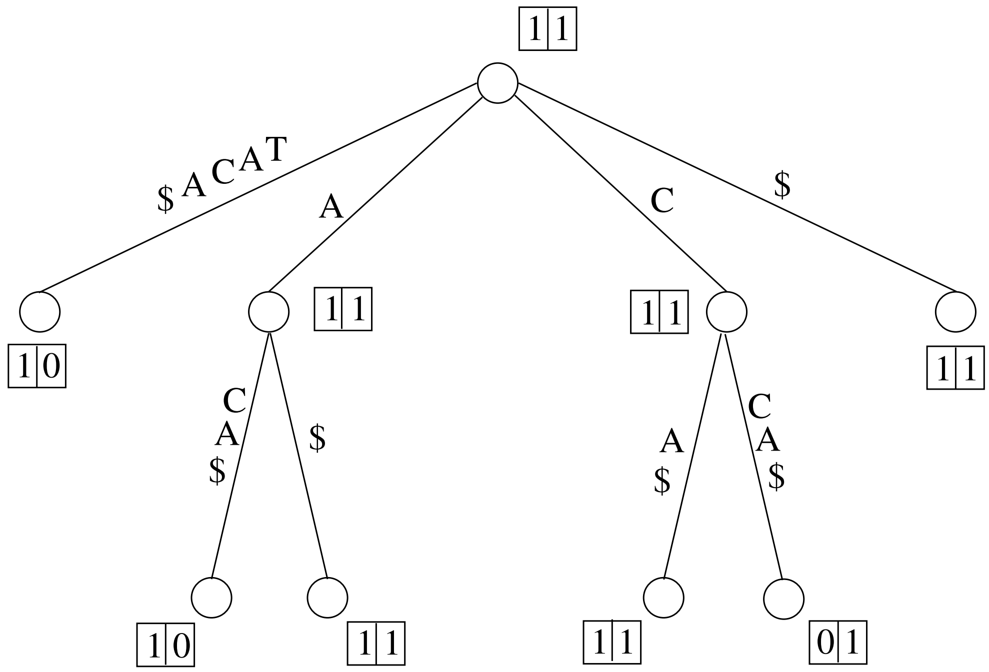

FIG. 4.7 – Un arbre des suffixes généralisé des séquences "TACA" et "CCA".
À côté de chaque noeud figure le tableau de bits (*Colors*) indiquant quelles sont les séquences contenant les suffixes épelés dans le sous-arbre du noeud : le premier bit fait référence à la séquence "TACA", le second à la séquence "CCA".

4.3.3 Pseudo-code

Comme précédemment, la fonction ÉpelleModèles décrite ici est appelée initialement avec $l = 0$, $m = \varepsilon$ et $occ_m = \varnothing$.

Algorithme B

```
ÉpelleModèles(Longueur $l$, Modèle $m$, Occurrences $occ_m$)
Si $k_{min} \leq l \leq k_{max}$ Alors
  AfficheModèle($m, occ_m$)
FinSi
Si $l < k_{max}$ Alors /* doit-on étendre le modèle ? */
  Pour tout symbole $\alpha \in \Sigma$ Faire
    $occ_{m\alpha} = \emptyset$
    $Colors_{m\alpha}$ est initialisé à 0

    Pour tout couple $(x, x_{err}) \in occ_m$ Faire
      Si $x$ est un noeud de l'arbre Alors
        Pour tout symbole $\beta \in \Sigma$ tel qu'il existe une transition
          par $\beta$ partant de $x$ vers un (pseudo-)noeud $x'$ Faire
          Si $\alpha \neq \beta$ Alors $x'_{err} = x_{err} + 1$
          Sinon $x'_{err} = x_{err}$
          FinSi
          Si $x'_{err} \leq e$ Alors
            Ajouter le couple $(x', x'_{err})$ à $occ_{m\alpha}$
          FinSi
        FinPour
      Sinon /* $x$ est alors un pseudo-noeud */
        Soit $\beta \in \Sigma$ l'unique transition possible par $x$ et menant à $x'$
        Si $\alpha \neq \beta$ Alors $x'_{err} = x_{err} + 1$
        Sinon $x'_{err} = x_{err}$
        FinSi
        Si $x'_{err} \leq e$ Alors
          Ajouter le couple $(x', x_{err})$ à $occ_{m\alpha}$
        FinSi
      FinSi
    FinPour

    $Colors_{m\alpha} = \bigcup_{x \in occ_{m\alpha}} Colors_x$
    Si le nombre de bits à 1 dans $Colors_{m\alpha} \geq q$ Alors
      ÉpelleModèles($l + 1, m\alpha, occ_{m\alpha}$)
    FinSi
  FinPour
FinSi
```

4.3.4 Complexités

Les complexités de cet algorithme sont les suivantes :

Lemme 5 *(M.F. Sagot [Sagot, 1998])*
Considérant l'extraction sur un alphabet de taille constante de modèles de longueur fixée k avec e substitutions au plus, dans N séquences de longueur n, l'algorithme produit tous les modèles valides de la longueur requise en temps $\mathcal{O}(nN^2\mathcal{V}(e,k))$ et en espace $\mathcal{O}((k+N)nN)$.

Preuve : Le coût de construction de l'arbre des suffixes de N séquences de longueur n est en $\mathcal{O}(nN)$ en temps et en espace.

L'insertion de tableaux de bits à chaque noeud entraîne un surcoût de complexité de l'arbre en espace. Si w est la taille d'un mot machine, l'espace requis pour le tableau de bits d'un noeud est en $\mathcal{O}(N/w)$ et il y a au plus $2nN$ noeuds dans l'arbre. L'espace requis pour stocker les occurrences d'un modèle et de tous ses préfixes à un instant donné du déroulement de l'algorithme ne peut excéder knN. Nous obtenons donc une complexité totale en espace en $\mathcal{O}(nN + 2nN^2/w + knN) = \mathcal{O}((k+N)nN)$.

Les surcoûts en temps proviennent de la création et de l'utilisation des tableaux de bits. Le calcul des tableaux de bits se fait lors d'un parcours en profondeur de l'arbre après construction. Le tableau de bits d'un noeud est déduit de l'union des Σ tableaux de bits situés au bout de chacune de ses transitions. Pour chaque noeud, on a donc un coût en $\mathcal{O}(\Sigma N/w) = \mathcal{O}(N)$, et donc un surcoût total à la construction de l'arbre en $\mathcal{O}(nN^2)$ pour l'ensemble des $2nN$ noeuds sur un alphabet fixé.

Durant l'exécution de l'algorithme, l'opération la plus coûteuse est celle de la ligne 31 qui consiste à calculer le tableau de bits du modèle en cours (m_α) en faisant l'union des tableaux de bits de chacune de ses occurrences, ce qui est réalisé en $\mathcal{O}(N/w) = \mathcal{O}(N)$ pour chaque noeud-occurrence. Puisque, comme dans l'algorithme précédent, $\mathcal{V}(e,k)$ est une borne supérieure du nombre de visites que peut recevoir chacun des noeuds de l'arbre, la complexité en temps est en $\mathcal{O}(nN^2\mathcal{V}(e,k))$.□

Pour cet algorithme comme pour le précédent, dans le cas d'un alphabet non fixé, le cardinal de l'alphabet apparaîtrait dans la construction de l'arbre, avec un facteur $\log|\Sigma|$ en temps et un facteur $|\Sigma|$ en espace, et dans le déroulement de l'algorithme : le voisinage $\mathcal{V}(e,k)$ d'un modèle croît exponentiellement avec ce cardinal.

4.4 Discussion

Cet algorithme présente une façon efficace d'éviter le parcours de l'intégralité de l'espace des solutions sans pour autant faire appel à une heuristique. Il est linéaire en la taille des séquences traitées.

En théorie, l'algorithme est exponentiel en le nombre de substitutions autorisées : le nombre de mots $\mathcal{V}(e,k)$ dans le voisinage d'un mot de longueur k avec e substitutions est majoré par $k^e|\Sigma|^e$. Cependant, la complexité dans le pire des cas ne permet pas d'exprimer correctement les avantages de cette méthode. Il est en effet évident que tous les noeuds de l'arbre ne recevront pas $\mathcal{V}(e,k)$ visites : plus le modèle est long, moins son voisinage sera complet. Ce sont surtout aux modèles courts, traités en haut de l'arbre, que s'applique ce facteur.

En pratique, lors de la descente de l'arbre virtuel des modèles, toutes les occurrences du modèle courant et de ses préfixes sont accumulées. Quand le nombre de substitutions augmente, la profondeur sur laquelle toutes les occurrences de l'arbre des suffixes appartiennent à tous les modèles augmente également. Par exemple, pour des valeurs correspondant à celles utilisées dans le cadre d'une extraction de promoteurs dans des séquences d'ADN ($k = 10, e = 2, |\Sigma| = 4$), tout modèle de longueur 2 en cours d'extension a nécessairement $|\Sigma|^2 = 16$ noeuds-occurrences (sur un jeu de séquences de taille suffisante) et est donc valide. Passés ces deux niveaux où toutes les occurrences trouvées appartiennent au modèle, la proportion d'occurrences appartenant à un modèle décroît. Mais si on extrait des modèles plus longs avec la même proportion de substitutions (k=50, e=10), le « pic » au niveau duquel toutes les occurrences d'un niveau de l'arbre appartiennent à n'importe quel modèle se situe maintenant à la profondeur 10 et peut atteindre $|\Sigma|^{10} = 1.048.576$ noeuds-occurrences (en fonction du jeu de données fourni).

Le champ d'action de cet algorithme se limite donc à l'extraction de modèles qui ne sont pas trop longs et assez conservés [6], ce qui correspond bien aux critères structurels des sites que nous souhaitons extraire. En effet, les sites de fixation auxquels nous nous intéressons ont généralement une longueur de l'ordre de la dizaine de nucléotides et une conservation raisonnable.

Tel qu'il est décrit dans l'article d'origine, cet algorithme est un filtre combinatoire qui extrait d'un jeu de séquences des modèles respectant certains critères structurels. Du point de vue du problème d'inférence que nous nous sommes posé, il est cependant incomplet pour plusieurs raisons :

- sa structure de donnée est trop lourde parce que trop générale : un arbre des suffixes permet de détecter la présence d'un facteur de longueur quelconque dans une séquence. Or les objets que nous cherchons à détecter sont soumis à des contraintes de longueur, ce dont ne tient pas compte cette structure de données. Marie-France Sagot suggérait, au terme de l'article dans lequel était présenté cet algorithme, de modifier la structure d'arbre en ce sens ;
- l'algorithme n'utilise pas les particularités de certains sites auxquels nous nous intéressons. En particulier, il ne tient pas compte d'éventuelles associations de plusieurs sites distants. Les critères structuraux pris en compte sont relativement simples (longueur, substitutions, quorum) et trop faibles pour qu'une inférence soit basée sur ces seuls critères comme nous le souhaitons. Notre volonté d'appliquer des critères structurels forts comme base d'une inférence implique des modifications importantes des modèles manipulés et des algorithmes d'extraction, afin notamment de réaliser l'extraction de modèles associés entre eux par des contraintes de distance. L'extension

[6] en ne perdant pas de vue que le nombre de substitutions e d'un modèle implique une distance d'au plus $2e$ entre ses occurrences.

des caractéristiques des modèles doit leur apporter davantage de spécificité.
Le problème d'inférence de sites de fixation nécessite d'autre part qu'on applique des critères de signification aux modèles. Comme nous l'expliquions à la fin du chapitre précédent, notre choix est de réaliser cette étape après que les critères structurels aient été appliqués, en post-traitement des données produites par l'extraction. Il reste donc également à rechercher et implanter de telles méthodes d'évaluation de modèles simples, mais aussi de modèles supportant des critères structurels plus forts, tels que nous les décrirons dans le chapitre qui suit.

Le chapitre suivant est consacré aux travaux théoriques que nous avons effectués à partir de cet algorithme sur les modèles simples. Auparavant, notre première tâche à été de l'implanter tel quel afin de vérifier son efficacité en pratique, expérience qui s'est avérée concluante (et dont les résultats figurent dans [Marsan, 1998]). Nous avons alors travaillé à définir des modèles étendus respectant des critères structuraux plus forts, plus proches des sites que nous souhaitons détecter, et à concevoir des algorithmes efficaces pour les extraire.

L'outil final que nous avons conçu, baptisé SMILE, sera décrit dans le chapitre 6.

Chapitre 5

Extraction de modèles structurés

Ce chapitre traite des travaux effectués à partir de l'algorithme d'extraction de modèles simples communs à plusieurs séquences décrit dans le chapitre précédent. Nous basant sur les spécificités du problème de l'inférence de sites de fixation, nous définissons une représentation qui permet de modéliser des associations entre plusieurs sites. Nous développons ensuite plusieurs algorithmes adaptés à cette représentation, ainsi qu'un outil baptisé SMILE. Ce chapitre contient toutes les bases théoriques à l'élaboration de SMILE, qui est décrit dans le chapitre suivant.

La première section concerne les modifications apportées aux représentations et structures de données de l'algorithme de Marie-France Sagot :

- à l'arbre des suffixes afin de l'alléger en tenant compte des spécificités de longueur des modèles que nous recherchons ;
- aux modèles eux-mêmes, afin de mieux décrire les sites de fixation que nous souhaitons modéliser et détecter. Nous introduisons une nouvelle notion : le modèle structuré.

La section 5.2 étend l'algorithme original à l'extraction de ces nouveaux modèles structurés. Plusieurs approches sont décrites visant à obtenir un algorithme efficace pour détecter des sites de fixation composés d'un nombre quelconque de motifs séparés par des *spacers* de longueur variable, et éventuellement très longs. De par l'algorithme dont ils s'inspirent, ceux que nous développons permettent en outre de prendre en considération les substitutions, d'intégrer sans difficulté importante la gestion des *gaps*, de traiter les *jokers* et de générer des modèles sur un alphabet dégénéré.

Enfin, nous proposons dans la section 5.3 une optimisation permettant de réduire encore le nombre de modèles potentiels visités. L'algorithme de Marie- France Sagot exploitait une propriété des modèles pour éviter de parcourir l'intégralité de l'espace des solutions. De même, nous prenons en compte une nouvelle propriété qui permet de réduire significativement ce parcours.

5.1 Modifications apportées aux représentations et structures

5.1.1 Un arbre plus léger

Les travaux théoriques et pratiques présentés dans cette section ont été réalisés par Julien Allali, *alors étudiant en Maîtrise d'informatique à l'université de Marne-la-Vallée, au cours de son stage à l'Institut Pasteur.*

L'arbre des suffixes a pour principal inconvénient d'être assez coûteux en mémoire. Bien que sa complexité en espace soit linéaire, la constante associée est assez importante et, dans notre cas, augmentée par le poids parfois conséquent des tableaux de bits se trouvant dans chaque noeud, et qui sont de taille proportionnelle au nombre de séquences traitées.

Cette structure a la propriété de permettre la détection de n'importe quel mot de la (ou des) séquences(s) stockées en un temps linéaire en la longueur de ce mot. Or cette propriété nous offre bien plus qu'il ne nous en faut puisque, lors d'une extraction, nous définissons entre autres la longueur maximale [1] des modèles à extraire. Nous n'avons donc nul besoin d'un arbre de tous les suffixes : lors d'une extraction de modèles dont la longueur est comprise entre k_{min} et k_{max}, un arbre de tous les facteurs de longueur comprise entre ces deux bornes suffit. La figure 5.1 illustre un tel arbre des facteurs.

Dans l'arbre des suffixes d'une séquence s, la profondeur maximale est égale à la longueur du plus long suffixe de la séquence, c'est-à-dire $|s|$, celle de la séquence elle-même. On y dénombre autant de suffixes que de positions de départ pour ceux-ci, soit $|s| + 1$ si l'on inclut le mot vide. Tous les chemins se terminent par un symbole $'\$'$.

Dans l'arbre des facteurs de longueur l, la profondeur maximale est évidemment l et le nombre de facteurs décrits vaut au plus $|s| - l + 1$. Les chemins des facteurs de longueur l ne se terminent pas par $'\$'$, ce symbole d'arrêt étant inutile si l'on se contente de rechercher dans l'arbre les facteurs n'excédant pas la longueur annoncée à la construction.

L'implantation sur laquelle nous nous sommes basés est celle d'Ukkonen ([Ukkonen, 1995]). La difficulté rencontrée lors de la construction d'un tel arbre des facteurs était de préserver la linéarité en temps de la construction. On peut bien sûr choisir de tronquer l'arbre après sa construction, par un parcours en profondeur, mais :

- l'implantation d'Ukkonen est « en ligne » : à l'étape i de la construction (pour i compris entre 1 et $|s|$), l'arbre suffixe de la séquence $s[1...i]$ est construit. En fabriquant l'arbre des facteurs par troncature de l'arbre des suffixes après construction, on perdrait ce caractère « en ligne » ;
- tronquer l'arbre des suffixes pour construire l'arbre des facteurs implique que la construction de cet arbre des facteurs a un coût en espace égal à celle de l'arbre des suffixes pendant au moins une partie de la construction.

[1] dans le cas où l'on ne souhaite pas fixer de longueur maximale lors d'une extraction, les améliorations décrites dans cette partie ne s'appliquent pas et l'arbre construit correspond à l'arbre des suffixes classique.

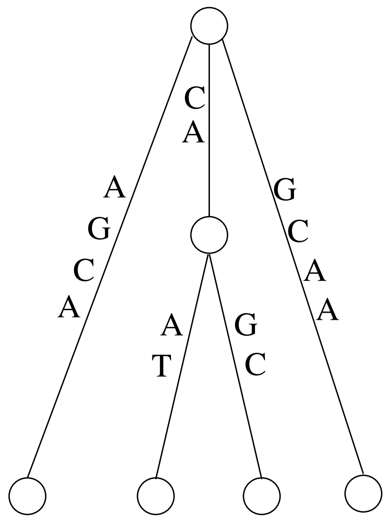

FIG. 5.1 – L'arbre des facteurs de longueur maximale 4 de la séquence "CAGCAAT" (à comparer avec l'arbre des suffixes de la même séquence, figure 4.4 page 85).

Pour éviter ces inconvénients, nous avons développé un algorithme spécifique dont on trouvera le détail dans [Allali, 2000]. Si les objectifs que nous nous étions fixés sont atteints (les linéarités en temps et en espace sont préservées, l'algorithme est « en ligne », et on ne construit pas l'arbre des suffixes), nous n'avons pas déterminé de meilleures bornes de complexités que celles de l'arbre des suffixes, qui permettraient d'évaluer les gains obtenus en fonction de la taille des facteurs stockés. Cela reste un problème ouvert, qui sera discuté dans [Allali and Sagot, 2002].

En pratique, cette amélioration ainsi que d'autres optimisations faites par Julien Allali ont réduit l'espace nécessaire à l'arbre utilisé par SMILE pour traiter N séquences de longueur n d'un facteur allant de 3 à 10 dans les problèmes que nous traitons. La complexité en espace, bien que plus faible qu'auparavant, reste importante : on observe des rapports de l'ordre de $100nN$ (où n est la longueur moyenne des séquences et N leur nombre), ce qui permet toutefois l'utilisation de SMILE sur des jeux de séquences de l'ordre du million de bases sur une machine ayant suffisamment de mémoire vive.

Notons que la majorité du poids de cette structure provient des tableaux de bits. Il nous faudrait trouver une autre méthode de stockage de ces tableaux ou une autre façon d'obtenir l'information de couverture qu'ils contiennent. Cela fait partie des travaux restant à effectuer sur la structure de données. Parmi les solutions envisagées pour l'alléger, l'utilisation d'une table de hachage pour atteindre les préfixes d'une longueur donnée et les sous-arbres correspondants pourrait s'avérer efficace (voir pour cela [Allali, 2000]).

5.1.2 Modèles structurés : plus proches des sites biologiques

Les modèles, dits « simples », utilisés dans l'algorithme original sont considérés comme inadéquats pour décrire les spécificités des sites de fixation. Les contraintes structurelles imposées par ce type de représentation sont trop faibles pour décrire efficacement des sites de fixation : au terme d'une extraction de modèles simples, on obtient souvent un grand nombre de modèles valides, preuve que les contraintes structurelles imposées sont trop souvent respectées. Seule l'estimation de la signification de ces modèles permet alors de distinguer ceux qui pourraient être pertinents.

Les observations et constats faits dans les chapitres précédents nous amènent à définir une nouvelle représentation pour les sites de fixation, dont les critères structurels sont renforcés afin de la rendre plus spécifique. Ce sont les associations entre sites de fixation fréquemment rencontrées chez certains organismes qui motivent l'élaboration de cette structure. Nous avons noté dans le chapitre 1 que les promoteurs et sites de régulation étaient dans certains cas sujets à des associations à des distances contraintes. Du fait de la structure spatiale des facteurs de transcription qui viennent s'y fixer et des éventuelles interactions entre ces facteurs, ces contraintes de distances entre sites peuvent de plus avoir un caractère plus ou moins rigide. La représentation que nous définissons ici permet d'exprimer cette flexibilité. Elle consiste en l'association ordonnée de plusieurs modèles simples, réunis pour former un *modèle structuré*.

Un modèle structuré est soumis à trois types de contraintes structurelles :
- locales, propres à chaque modèle simple qui le compose ;
- globales, propres à l'ensemble de la représentation. Elles permettent d'exprimer des corrélations entre les différents modèles simples qui la constituent ;
- de saut entre chaque paire de modèles simples consécutifs.

Définition 9 (Modèle structuré)
Un modèle structuré *est un ensemble composé de plusieurs modèles simples (qu'on nommera* boîtes *ou* blocs*) séparés par des intervalles de distances contraints.*
Formellement, un modèle structuré composé de p boîtes est une paire (m, d) telle que :
- *m est un vecteur contenant p boîtes $(m_1, \ldots, m_p)$ (où $m_i \in \Sigma^+$) ;*
- *d est un vecteur contenant $p-1$ triplets $((d_{min_1}, d_{max_1}, \delta_1), \ldots, (d_{min_{p-1}}, d_{max_{p-1}}, \delta_{p-1}))$ qui indiquent les contraintes de saut entre deux boîtes consécutives (avec pour tout i d_{min_i}, $d_{max_i} \in \mathbb{N}, \delta_i \in \mathbb{N}^*$ et $d_{min_i} \leq d_{max_i}$).*

Le triplet $(d_{min_i}, d_{max_i}, \delta_i)$ du vecteur d qui définit un modèle structuré (m, d) indique les modalités du saut entre la boîte i et la boîte $i+1$: d_{min_i} et d_{max_i} sont les bornes du saut et δ_i le « rayon » de chaque fenêtre de saut à considérer entre ces bornes.

Notre volonté était d'ajouter une contrainte de distance souple entre des modèles simples ; pour ce faire, un modèle structuré exprime des fourchettes $[d_{min_i}; d_{max_i}]$ dans lesquelles doit être comprise la longueur du *spacer* qui sépare les boîtes i et $i+1$. L'ajout de cette unique contrainte entre boîtes permet de modéliser des associations de sites pour lesquelles on connaîtrait une fourchette de longueur des *spacers* mis en jeu. Par exemple, dans le cas des promoteurs de *E.coli* présentés dans l'introduction, on utiliserait une fourchette $[15; 19]$ entre les deux boîtes étant donné que les occurrences des deux promoteurs peuvent être séparées par un *spacer* contenu dans cet intervalle.

Dans l'optique d'une inférence où l'on connaît moins de détails sur les distances entre boîtes, nous avons introduit la notion de *delta*, qui est en fait un « rayon » à considérer pour définir des sous-intervalles, de largeur $2\delta + 1$, dans l'intervalle principal. Pour qu'un modèle structuré soit valide, il faut que le quorum soit atteint pour au moins un de ces sous-intervalles, et ce entre chaque paire de boîtes successives du modèle.

Les modèles structurés contiennent donc intrinsèquement des propriétés aidant à leur inférence sur des critères souples.

Définition 10 (Modèle structuré valide)
Étant donnés un jeu de N séquences $s_1, \ldots, s_N$ et un quorum q, un modèle structuré (m, d) composé de p boîtes est dit valide si on peut trouver $nb \geq q$ « trains » d'occurrences $u_1, \ldots, u_p$ de $m_1, \ldots, m_p$ dans au moins q séquences distinctes, tels que :
- *chacune des occurrences d'un « train » $u_1, \ldots, u_p$ appartient à la même séquence ;*
- *quel que soit i, $1 \leq i \leq (p-1)$, deux occurrences successives u_i et u_{i+1} respectent les contraintes de saut imposées, à savoir : il existe un $d_i \in [d_{min_i} + \delta_i; d_{max_i} - \delta_i]$ tel que la distance qui sépare la fin de u_i et le début de u_{i+1} soit égale à $d_i \pm \delta_i$;*
- *quel que soit i, $1 \leq i \leq (p-1)$, d_i est le même pour les nb « trains » d'occurrences.*

La figure 5.2 présente un modèle structuré composé de deux boîtes et montre le processus de validation ou d'invalidation de ce modèle en fonction des couvertures recueillies pour chaque fenêtre de l'intervalle de saut.

Comme auparavant, chaque boîte possède son nombre de substitutions propre e_i, préalablement choisi ; c'est une contrainte locale. Un modèle structuré possède de plus un nombre de substitutions maximal global, noté e_{global}. De même qu'un nombre maximum de substitutions pour un modèle simple introduit, de façon limitée, des possibles interactions entre les positions du modèle, ce nombre global de substitutions établit des interactions limitées entre les différentes boîtes d'un modèle structuré.

Une instance simplifiée des modèles structurés consiste à évincer les deltas. On considère dans ce cas les modèles structurés tels que présentés dans la définition 9, mais dans lesquels tous les deltas (δ) ont été otés des vecteurs. De la même manière, la validation de tels modèles structurés sans deltas est une instance simplifiée de celle vue précédemment :

Définition 11 (Modèle structuré valide sans delta)
Étant donnés un jeu de N séquences $s_1, \ldots, s_N$ et un quorum q, un modèle structuré (m, d) composé de p boîtes est dit valide si on peut trouver $nb \geq q$ « trains » d'occurrences $u_1, \ldots, u_p$ de $m_1, \ldots, m_p$ dans au moins q séquences distinctes, tels que :
- *chacune des occurrences d'un « train » $u_1, \ldots, u_p$ appartient à la même séquence ;*
- *quel que soit $1 \leq i \leq (p-1)$, deux occurrences successives u_i et u_{i+1} respectent les contraintes de saut imposées, à savoir : la distance qui sépare la fin de u_i et le début de u_{i+1} est inclue dans l'intervalle $[d_{min_i}; d_{max_i}]$.*

Dans l'exemple de la figure 5.2, les quatre dernières occurrences appartiennent au modèle sans delta, et ce dernier est donc valide.

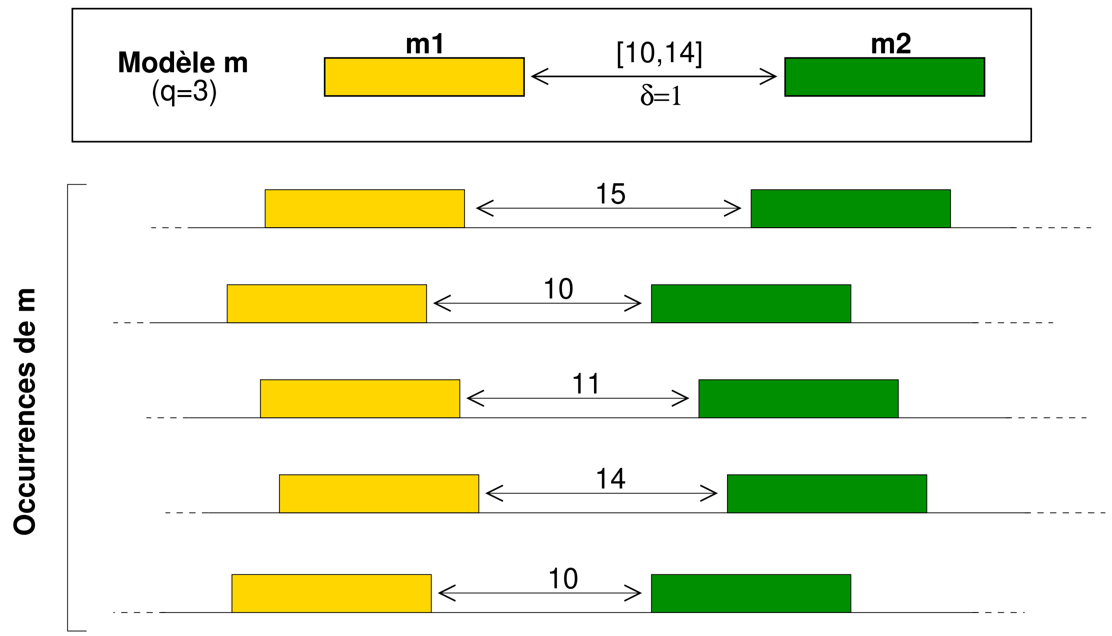

FIG. 5.2 – Validation d'un modèle structuré.
Le modèle structuré m est composé de deux boîtes $m1$ et $m2$, le saut entre ces deux boîtes est compris entre $d_{min} = 10$ et $d_{max} = 14$, avec un $\delta = 1$. Le quorum q est fixé à 3 séquences. Nous ne considérons ici que les critères de saut, et omettons donc les nombres de substitutions propres à chaque boîte. Ce faisant, on trouve 5 occurrences des modèles simples $m1$ et $m2$ dans 5 séquences distinctes. Le modèle structuré est-il valide ? Si oui, pour quel(s) intervalle(s) de saut ?
L'intervalle de saut global $[10; 14]$ et un $\delta = 1$ impliquent qu'il y a trois sous-intervalles de saut à tester : 11 ± 1, celui-ci est valide car il contient trois occurrences (2, 3 et 5) ; 12 ± 1 n'est pas valide car il ne contient qu'une occurrence (la troisième) ; de même pour 13 ± 1 qui ne contient que la quatrième occurrence. Le modèle est donc valide pour l'intervalle $[10; 12]$ uniquement.
Dans une instance simplifiée, sans deltas, on considérera que le modèle m recueille simplement les quatre dernières occurrences, et est donc valide.

Des modèles aux propriétés étendues ont été définis, permettant de gérer des associations entre modèles simples. En prenant en compte des corrélations de ce type (et éventuellement d'autres contraintes structurelles), nous souhaitons obtenir une représentation plus spécifique (car un modèle structuré décrit mieux les sites que nous recherchons) que les modèles simples, et permettant de développer des algorithmes plus efficaces (car l'espace des solutions est réduit par ces contraintes structurelles plus fortes) que l'algorithme original. Ces critères structurels forts nous permettraient de réaliser une inférence de sites fondée sur ces critères, de manière exacte. L'utilisation de critères de signification *a posteriori*, bien qu'inévitable, pourrait alors être rendue véritablement secondaire.

5.2 Algorithmes d'extraction de modèles structurés

Dans cette partie, nous développons plusieurs algorithmes combinatoires exacts permettant d'extraire des modèles structurés de façon efficace. Ces travaux ont été initialement présentés dans [Marsan and Sagot, 2001].

Comme à notre habitude, nous traiterons d'abord une instance simplifiée du problème à résoudre avant de nous attaquer au cas le plus complexe. Pour ce faire, nous ne considérerons pour commencer que des modèles composés de deux boîtes et sans delta. Puis, nous étendrons progressivement cet algorithme simplifié à des versions à p boîtes acceptant les deltas.

Les outils que nous avons implémentés à partir des algorithmes de ce chapitre et qui seront décrits dans le chapitre suivant gèrent des modèles avec ou sans deltas.

Étant données N séquences $s_1, \ldots, s_N$ et un quorum $q \leq N$, nous posons les problèmes suivants :
Problème 1 : trouver tous les modèles structurés composés de deux boîtes, sans delta, notés $((m_1, m_2), (d_{min_1}, d_{max_1}))$, qui sont valides ;
Problème 2 : trouver tous les modèles structurés composés de p boîtes, sans deltas, notés $((m_1, \ldots, m_p), ((d_{min_1}, d_{max_1}), \ldots, (d_{min_{p-1}}, d_{max_{p-1}})))$, qui sont valides ;
Problème 3 : trouver tous les modèles structurés composés de deux boîtes, avec delta, notés $((m_1, m_2), (d_{min_1}, d_{max_1}, \delta_1))$, qui sont valides ;
Problème 4 : trouver tous les modèles structurés composés de p boîtes, avec deltas, notés $((m_1, \ldots, m_p), ((d_{min_1}, d_{max_1}, \delta_1), \ldots, (d_{min_{p-1}}, d_{max_{p-1}}, \delta_{p-1})))$, qui sont valides.

L'omission du delta n'a pas uniquement pour but de simplifier le problème : quand on ne connaît pas précisément son intervalle de saut, le delta permet de détecter un modèle structuré en précisant seulement un intervalle global dans lequel cet intervalle de saut est susceptible d'être situé, et son rayon (le delta). Dans le cas d'un modèle dont on connaît *a priori* l'intervalle de saut, l'utilisation du delta s'avère inutile.

Afin de simplifier les notations, nous considérerons (sauf mention contraire) pour un modèle à p boîtes que chacune d'elles possède les mêmes critères locaux (même longueur k, même nombre maximal de substitutions e), et que chaque intervalle possède les mêmes critères de saut (pour chaque intervalle de saut $1 \leq i \leq (p-1)$ les d_{min_i}, d_{max_i} et éventuellement les δ_i ont les mêmes valeurs d_{min}, d_{max}, δ). Par exemple, les modèles du problème 2 sont alors de la forme $((m_1, \ldots, m_p), ((d_{min}, d_{max}), \ldots, (d_{min}, d_{max})))$. Par ailleurs, nous considérons que le jeu de données est composé de séquences de longueur n sur un alphabet Σ, à partir desquelles on construit l'arbre des suffixes (ou des facteurs) généralisé $\mathcal{A}$.

Des solutions aux problèmes 1, 2, 3 et 4 sont présentées dans les sections 5.2.1, 5.2.2, 5.2.3 et 5.2.4 respectivement. La section 5.2.5 présente les dernières mises au point qui finalisent ces algorithmes pour les rendre génériques, en y ajoutant des critères globaux.

5.2.1 Modèles composés de deux boîtes à intervalle de saut connu (Problème 1)

Nous présentons dans cette section plusieurs algorithmes permettant de résoudre le problème 1 (la figure 5.3 rappelle les caractéristiques du modèle structuré à extraire). Ces algorithmes ont pour base l'algorithme d'extraction de modèles simples communs à plusieurs séquences, présenté au chapitre précédent.

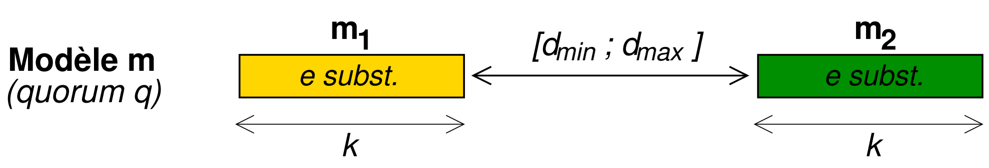

FIG. 5.3 – Caractéristiques des modèles à extraire dans le problème 1.

Dans la section 5.2.1.1, cet algorithme de base est d'abord étendu de façon naïve à l'extraction de modèles structurés. Puis, nous présentons dans les sections 5.2.1.2 et 5.2.1.3 deux algorithmes nouveaux qui impliquent de plus profondes modifications de la version originale.

Notons que l'algorithme 2 décrit dans la section 5.2.1.3 et qui introduit la notion d'arbre dynamique, nécessite de disposer des liens-suffixes créés lors de la construction de l'arbre des suffixes et inutilisés jusqu'alors.

5.2.1.1 Approche naïve

Description

Une façon naïve de résoudre le problème 1 consiste à utiliser l'algorithme d'extraction de modèles simples pour extraire et stocker tous les modèles de longueur k avec e substitutions respectant un quorum q. Puis, il suffit alors de rechercher parmi les modèles simples obtenus les couples qui peuvent s'associer pour former un modèle structuré du type de ceux que nous souhaitons extraire.

Pour ce faire on construit un tableau de taille nN où chaque cellule correspond à une position dans les séquences traitées. Pour chaque cellule, on stocke la liste des modèles ayant une occurrence à cette position. On dispose d'autre part de la sortie de l'extraction, contenant l'ensemble des modèles et leurs occurrences.

Pour chaque position i du tableau et pour chaque modèle m_1 trouvé, on va rechercher si les modèles se trouvant en $i+d_{min}+1$ à $i+dmax+1$ peuvent constituer une seconde boîte pour m_1. Les couples de modèles dont le nombre de couples d'occurrences relevé atteint le quorum q requis sont validés comme modèles structurés.

Complexités
L'extraction de tous les modèles de longueur k nécessite un temps $nN^2\mathcal{V}(e,k)$ et un espace $(k+N)nN$ (voir le lemme 5 page 95). Il faut ajouter à cela l'espace du tableau de positions : chaque cellule correspond à une position parmi nN, et chaque position peut recevoir dans le pire des cas $\mathcal{V}(e,k)$ modèles différents (nombre de modèles dans le voisinage desquels une occurrence peut se trouver). Ce tableau a donc un coût en espace en $\mathcal{O}(nN\mathcal{V}(e,k))$. Stocker l'ensemble des modèles produits lors de l'extraction et leurs occurrences coûte le même espace. La complexité totale en espace est donc de $\mathcal{O}((k+N)nN + 2nN\mathcal{V}(e,k))$.

Une fois effectuée l'extraction de modèles simples, on parcourt les nN positions du tableau, et les $\mathcal{V}(e,k)$ modèles au plus se trouvant à chaque position. Pour chaque modèle m_1 se trouvant à la position i, on parcourt l'ensemble des modèles se trouvant aux positions $i+d_{min}+1$ à $i+d_{max}+1$. On rencontre au plus $\Delta\mathcal{V}(e,k)$ modèles (avec $\Delta = d_{max} - d_{min} + 1$). Pour chaque modèle m_2 rencontré, on parcourt simultanément les listes d'occurrences (triées) de m_1 et m_2 afin de compter le nombre de séquences où ils sont associées en respectant les critères requis. Cela prend un temps en nN (nombre maximum d'occurrences d'un modèle).

Au final, la complexité en temps du traitement des résultats de l'extraction est en :

$$\mathcal{O}(\underbrace{nN}_{\text{Positions}} \times \underbrace{\mathcal{V}(e,k)}_{\text{Modèles/position}} \times \underbrace{\Delta\mathcal{V}(e,k)}_{\text{Modèles associables}} \times \underbrace{nN}_{\text{Comparaison occurrences}}) = \mathcal{O}(\Delta(nN\mathcal{V}(e,k))^2)$$

La complexité de l'extraction est noyée dans cette dernière, on obtient donc une complexité totale en temps en $\mathcal{O}(\Delta(nN\mathcal{V}(e,k))^2)$.

5.2.1.2 Algorithme 1 : sauter dans l'arbre

Nous intégrons cette fois le concept de modèle structuré directement dans l'algorithme d'extraction, plutôt que de le faire par post-traitement comme dans la version naïve qui précède.

Description
Le principe général consiste à extraire la seconde boîte de la même façon que la première, en réalisant le saut qui les sépare directement dans l'arbre des suffixes. Ce saut permet d'atteindre les noeuds-occurrences correspondants aux débuts potentiels d'une seconde boîte à partir des occurrences d'une première boîte donnée.

Considérons l'étape de l'algorithme de base où un modèle m_1 de longueur k a été validé. À cet instant, on dispose de son jeu de noeuds-occurrences V_1 (ces noeuds se situant tous à la profondeur k de l'arbre $\mathcal{A}$). Nous souhaitons alors procéder à l'extraction de tous les modèles m_2 qui composent un éventuel modèle structuré $((m_1, m_2), (d_{min}, d_{max}))$ valide.

Cela se fait en deux étapes, illustrées dans la figure 5.4 :

- **(saut)** depuis chaque noeud v de V_1, on réalise un saut dans l'arbre $\mathcal{A}$ jusqu'à tous les noeuds correspondant à des débuts potentiels d'occurrences w d'un second modèle m_2 pour m_1. Ces descentes se font sans comparaisons, donc plus rapidement que les extensions habituelles. Le jeu d'occurrences V_2 ainsi créé contient tous les descendants de V_1 se situant entre les niveaux $k + d_{min}$ et $k + d_{max}$ inclus de $\mathcal{A}$ (chaque élément de V_1 engendre plusieurs éléments de V_2). On définit l'ensemble V_2 des noeuds-occurrences d'une seconde boîte par :
$$V_2 = \{(w, e_w = e_v)/\exists v \in V_1 \text{ } tel \text{ } que \text{ } d_{min} \leq niveau(w) - niveau(v) \leq d_{max}\}$$
Le but de cette manoeuvre est de nous amener dans les conditions qui permettront l'extension du modèle $((m_1, m_2), (d_{min}, d_{max}))$ avec $m_2 = \varepsilon$;
- **(extraction)** disposant de toutes les occurrences nécessaires dans V_2, on peut maintenant étendre le modèle structuré $((m_1, m_2 = \varepsilon), (d_{min}, d_{max}))$ avec ce jeu d'occurrences, et ce de la même manière que dans la règle 1 de l'algorithme original : on obtiendra un modèle structuré valide si l'on parvient à étendre le modèle m_2 à la longueur k requise en respectant le quorum q.

Par rapport à l'algorithme naïf, cet algorithme apporte une extraction contextuelle de la seconde boîte : on n'extrait pas toutes les secondes boîtes mais seulement celles qui peuvent s'associer à la première boîte extraite.

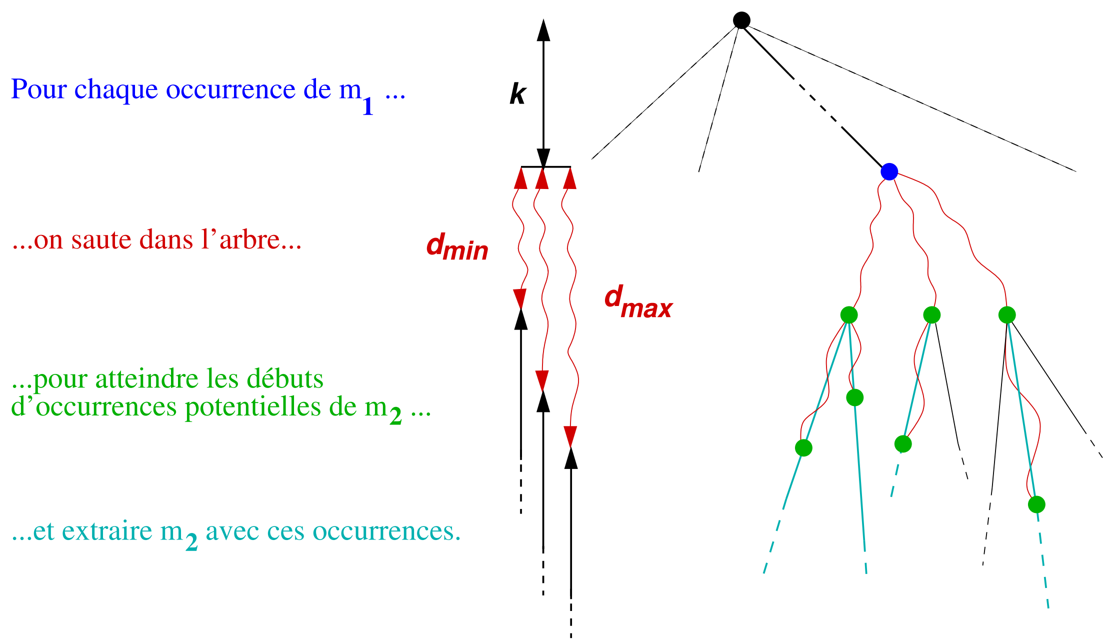

FIG. 5.4 – Algorithme 1 : le saut dans l'arbre des suffixes.

Notons que, si dans l'algorithme simple du chapitre précédent les occurrences d'un modèle avaient toujours la même longueur que ce modèle, ce n'est plus le cas ici, à partir du moment où l'on a réalisé le saut dans l'arbre des suffixes. En effet, lors de l'extension du premier symbole d'une seconde boîte, on dispose d'un jeu de noeuds-occurrences se situant aux niveaux $k + d_{min}$ à $k + d_{max}$ de l'arbre.

Comme conséquence des modifications apportées précédemment à la structure de données, nous pouvons maintenant profiter du fait que les noeuds-occurrences utilisés se situent à une profondeur qui n'excède pas le niveau $2k + dmax$. On limitera par conséquent la hauteur de l'arbre en construisant à la place de l'arbre des suffixes un arbre des facteurs n'excédant pas la longueur $2k + dmax$.

Pseudo-code

La procédure ÉpelleModèlesStruct suivante décrit les points principaux du nouvel algorithme. Elle est appelée initialement avec le modèle $m = ((m_1 = \varepsilon, m_2 = \varepsilon), (d_{min}, d_{max}))$ et le numéro de bloc en cours d'extension (1 pour débuter). La variable k est la longueur requise pour chaque bloc, $\mathcal{A}$ est l'arbre (des suffixes ou des facteurs) des séquences.

Algorithme 1

`ÉpelleModèlesStruct`(Modèle m, Bloc i)
Si $i = 2$ **Alors**
  Pour chaque noeud-occurrence v de m **Faire**
    Placer dans $DebutsPotentiels$ les fils de v se situant
      entre les niveaux $k + d_{min}$ et $k + d_{max}$
  FinPour
Sinon
  Placer v (la racine de l'arbre) dans $DebutsPotentiels$
FinSi

Pour chaque modèle valide m_i obtenu en descendant simultanément
  l'arbre des suffixes $\mathcal{A}$ et l'arbre virtuel des modèles
  à partir des noeuds-occurrences de $DebutsPotentiels$
  (algorithme B page 94) **Faire**
  Si $i = 1$ **Alors**
    `EpelleModelesStruct`($m = ((m_1, m_2 = \varepsilon), (d_{min}, d_{max}))$, $i + 1$)
  Sinon
    Reporter le modèle $m = ((m_1, m_2), (d_{min}, d_{max}))$ comme valide
  FinSi
FinPour

Complexités

Lemme 6
Pour extraire d'un jeu de N séquences de longueur n tous les modèles structurés de la forme $((m_1, m_2), (d_{min}, d_{max}))$, chaque bloc ayant une longueur k et un nombre de substitutions maximal e, la complexité en temps dans le pire des cas de l'algorithme 1 est en $\mathcal{O}(nN^2 + Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$ et sa complexité en espace est en $\mathcal{O}((2k + d_{max} + N)nN)$ sur un alphabet de taille constante.

Preuve : Il nous faut calculer le nombre de fois qu'un noeud-occurrence situé entre les niveaux $2k + d_{min}$ et $2k + d_{max}$ de l'arbre peut appartenir à un modèle structuré, et donc être visité.

On sait qu'au plus $\mathcal{V}(e,k)$ modèles simples peuvent contenir une occurrence de longueur k donnée. Une paire d'occurrences de longueur k peut donc, dans le pire des cas, appartenir à $\mathcal{V}^2(e,k)$ modèles structurés différents. Le nombre total de visites aux noeuds-occurrences susceptibles d'appartenir aux modèles structurés qui nous intéressent est par conséquent majoré par :

$$\sum_{i=d_{min}}^{d_{max}} n_{2k+i}\mathcal{V}^2(e,k) \leq min(2,\Delta)n_{2k+d_{max}}\mathcal{V}^2(e,k)$$

où n_i indique le nombre de noeuds de l'arbre $\mathcal{A}$ au niveau i (qui ne peut excéder nN, le nombre total de positions dans les séquences traitées). Le $min(2,\Delta)$ provient du fait que le nombre de noeuds entre les niveaux $2k + d_{min}$ et $2k + d_{max}$ de l'arbre ne peut excéder $2n_{2k+d_{max}}$.

Puisque chaque visite d'un noeud requiert, comme dans l'algorithme B, $\mathcal{O}(N)$ opérations au plus, la complexité en temps est donc en $\mathcal{O}(nN^2 + min(2,\Delta)Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$ (où le premier terme représente le temps de construction de l'arbre), ce qui équivaut, quel que soit Δ, à $\mathcal{O}(nN^2 + Nn_{2k+d_{max}}\mathcal{V}^2(e,k))$.

N'ayant pas de meilleure borne que celle de l'arbre des suffixes classique pour évaluer l'espace nécessaire à l'arbre des facteurs, et n'utilisant pas de structures supplémentaires, nous nous contentons de donner à cet algorithme la même complexité en espace que celle de l'algorithme B pour l'extraction de boîtes de longueur $2k+d_{max}$: $\mathcal{O}((2k+d_{max}+N)nN)$.$\square$

Cet algorithme représente une amélioration par rapport à l'algorithme naïf. On obtient en effet de meilleures complexités :
- en temps, puisqu'il est linéaire en la longueur des séquences au lieu d'être quadratique, et que le facteur Δ disparaît (Δ influe tout de même sur le nombre de noeuds atteints aux niveaux $2k + d_{min}$ à $2k + d_{max}$) ;
- en espace, puisque le facteur $\mathcal{V}(e,k)$ disparaît également, au profit d'un facteur $k + d_{max}$ moins coûteux.

L'étape de saut dans l'arbre fait en général augmenter rapidement le nombre d'occurrences du modèle en cours, l'extension de la seconde boîte doit donc se faire avec un nombre important d'occurrences. Anticipant sur l'extension de cet algorithme à l'extraction de modèles de plus de deux boîtes, on en déduit que plusieurs sauts consécutifs pourraient engendrer un nombre considérable d'occurrences à étendre. Cette remarque nous amène à l'élaboration d'un nouvel algorithme réalisant l'extraction différemment.

5.2.1.3 Algorithme 2 : un arbre dynamique des suffixes

Dans cet algorithme, les extractions de la seconde boîte se déroulent à partir du haut de l'arbre des suffixes et débutent toujours avec une seule occurrence du mot vide ε, comme débutent les extractions des premières boîtes. La structure de l'arbre est modifiée pour chaque première boîte extraite afin de prendre en compte son contexte dans l'extraction à effectuer pour la seconde boîte (d'où le nom d'arbre dynamique des suffixes).

Description
L'algorithme 2 procède initialement comme l'algorithme 1 : on commence par extraire un premier bloc de longueur k. Pour toutes les occurrences de chaque premier bloc m_1 de longueur k obtenu, comme dans l'algorithme 1, un saut est réalisé dans l'arbre $\mathcal{A}$. Cette fois cependant, ce saut n'a pour but que de recueillir certaines informations pour les ramener au niveau k de l'arbre. Ces informations sont ensuite utilisées pour transformer temporairement les k premiers niveaux de l'arbre $\mathcal{A}$ en un nouvel arbre $\mathcal{A}'$, qui n'exprime que les occurrences potentielles d'une seconde boîte de longueur k pour m_1. L'objectif de cette manoeuvre est de réaliser l'extraction des secondes boîtes m_2 depuis la racine de $\mathcal{A}'$ avec une unique occurrence du mot vide ε, comme on le fait pour extraire les premières boîtes. Une fois l'extraction de toutes les m_2 associées à m_1 réalisée, l'arbre $\mathcal{A}$ initial est rétabli et l'extraction d'une autre première boîte se poursuit récursivement.

Détaillons les opérations qui se déroulent entre la fin de l'extraction d'une première boîte et le début de l'extraction des secondes boîtes associées. Pour une première boîte m_1 donnée, le but de ces opérations est de transformer les k premiers niveaux de l'arbre $\mathcal{A}$ en un arbre $\mathcal{A}'$ de tous les facteurs de longueur k (des séquences $s_1, \ldots, s_N$) dont le début se trouve à une distance d_{min} à d_{max} de chacune des occurrences de m_1. L'arbre $\mathcal{A}'$ est en ce sens l'« union » de tous les sous-arbres de hauteur k enracinés aux noeuds-occurrences correspondants aux débuts potentiels d'occurrences de m_2 pour m_1.

Ces occurrences pour une seconde boîte sont déjà représentées dans le haut de l'arbre (comme tous les facteurs de la séquence), mais il nous faut éliminer temporairement les chemins de l'arbre menant aux autres occurrences. De plus, les tableaux de bits de ces occurrences de seconde boîte pour m_1 sont incorrects puisqu'ils indiquent leur couverture en séquences dans un contexte « première boîte », indépendamment du fait que seules celles qui sont des secondes boîtes potentielles nous intéressent. Pour construire le nouvel arbre $\mathcal{A}'$, il nous faut donc repérer dans $\mathcal{A}$ les facteurs de longueur k qui correspondent à des occurrences potentielles d'une seconde boîte de m_1 (et disqualifier temporairement les autres facteurs de longueur k), puis mettre à jour les tableaux de bits de ces occurrences.

Pour chaque boîte m_1 obtenue, on procède donc de la sorte :

- les tableaux de bits $Colors_z$ de tous les noeuds z se trouvant au niveau k de $\mathcal{A}$ sont stockés dans un tableau L de dimension n_k (nombre de noeuds au niveau k). Ce tableau permettra le rétablissement de $\mathcal{A}$ une fois l'extraction de la seconde boîte effectuée ;
- depuis chaque noeud-occurrence v de m_1, on effectue une descente à tous les noeuds-occurrences se trouvant entre les niveaux $2k+d_{min}$ et $2k+d_{max}$ de $\mathcal{A}$. Chaque noeud-occurrence w ainsi atteint correspond potentiellement à une fin d'occurrence pour une seconde boîte m_2 de m_1 (dans l'algorithme 1, on sautait aux débuts d'occurrences potentiels pour la seconde boîte) ;
- chacun de ces noeuds-occurrences w contient un tableau de bits $Colors_w$ qui indique les séquences dans lesquelles l'occurrence apparaît. C'est l'information qui nous intéresse et que nous remontons donc au niveau k de $\mathcal{A}$ en suivant le chemin des liens-suffixes qui part de w. Le noeud z ainsi atteint correspond à une occurrence potentielle de longueur k pour une seconde boîte de m_1 [2]. Si c'est la première visite de z, $Colors_z$ est copié à la place de $Colors_w$, sinon il y est ajouté (« OU » logique) ;
- une fois que cette descente/remontée a été effectuée pour toutes les occurrences v de m_1, les tableaux $Colors_z$ de chaque noeud z ayant été atteint sont propagés jusqu'à la racine de l'arbre, par addition comme précédemment. Les chemins ainsi parcourus forment le nouvel arbre $\mathcal{A}'$ des occurrences potentielles d'une seconde boîte de m_1. Les chemins de $\mathcal{A}$ non parcourus lors de cette étape ne font pas partie de $\mathcal{A}'$.

Pour donner une vue d'ensemble de cet algorithme, la figure 5.5 illustre l'extraction des deux boîtes d'un modèle structuré. L'extraction des secondes boîtes m_2 de m_1 (étape **(e)** de la figure 5.5) est effectuée dans $\mathcal{A}'$ de la même façon que dans l'algorithme B.

Une fois cette extraction de seconde boîte terminée, l'arbre $\mathcal{A}$ est rétabli en replaçant les tableaux de bits stockés dans L à leurs noeuds d'origine, et en propageant comme précédemment ces tableaux jusqu'à la racine de l'arbre.

Remarquons que, si nous parlons de deux arbres $\mathcal{A}$ et $\mathcal{A}'$ distincts, il n'y a en fait en mémoire qu'un seul arbre original $\mathcal{A}$, qui est temporairement modifié lors du déroulement de l'algorithme.

On note que les noeuds situés entre les niveaux $2k + d_{min}$ et $2k + d_{max}$ vont être fréquemment visités. Pour optimiser la remontée de ces noeuds vers les noeuds du niveau k (cette remontée se fait normalement en $k + i$ ($i \in [d_{min}; d_{max}]$) liens-suffixes successifs), l'arbre $\mathcal{A}$ est traité lors de sa construction, de façon à ce que le lien-suffixe qui part de chacun de ces noeuds aboutisse directement au niveau k. Ces liens-suffixes modifiés sont appelés des *liens-suffixes rapides*. La remontée d'un noeud de ces niveaux vers le niveau k par un lien-suffixe rapide prend donc un temps constant.

[2] le cheminement par les liens-suffixes nous a mené d'une occurrence de longueur $2k+i$ ($i \in [d_{min}; d_{max}]$) au suffixe de longueur k de cette occurrence, qui correspond à la deuxième boîte seule.

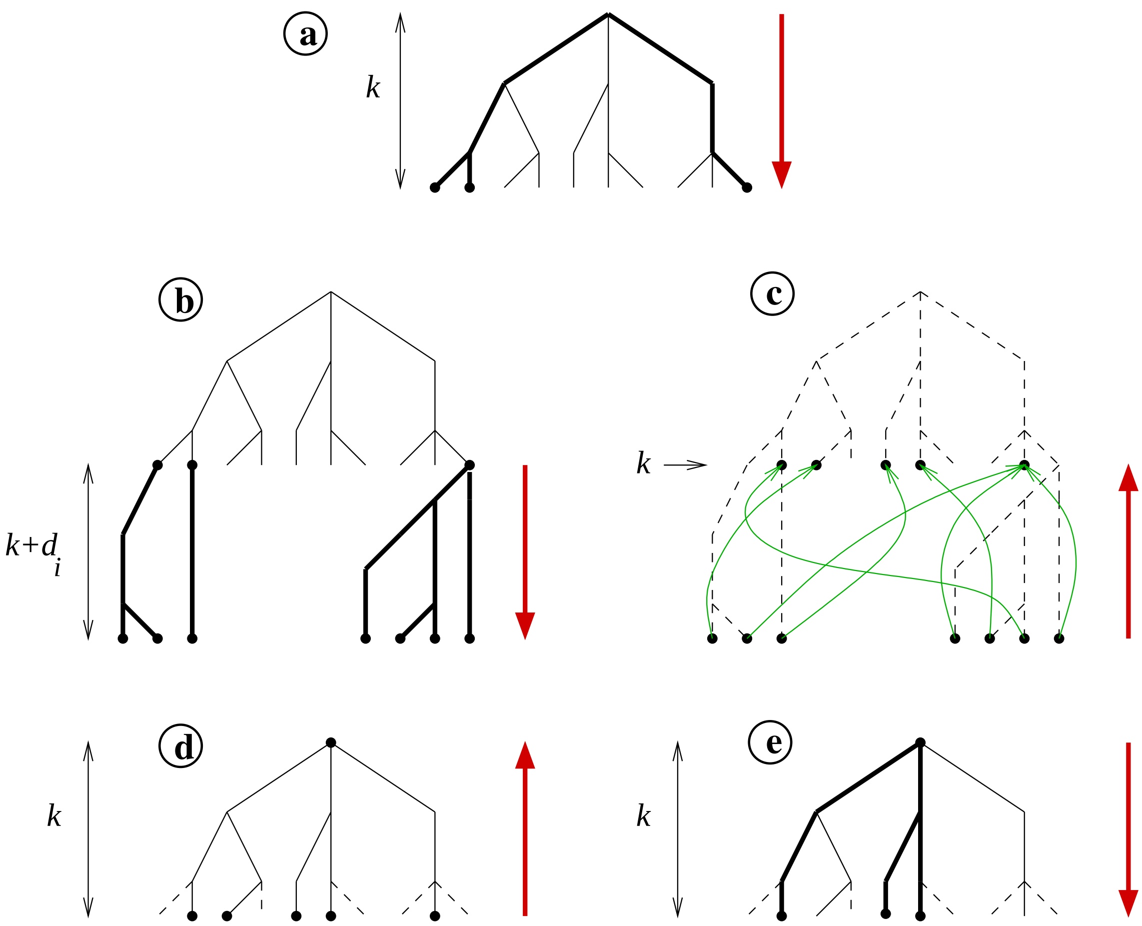

FIG. 5.5 – Algorithme 2 : extraction de deux boîtes dans l'arbre dynamique des suffixes.
a) un modèle m_1 est étendu (avec ses occurrences, en gras) au niveau k. b) à partir de ces occurrences, un saut est réalisé jusqu'à tous les noeuds se trouvant $k+i$ ($i \in [d_{min}; d_{max}]$) niveaux plus bas. c) de ces noeuds, les tableaux de bits sont remontés au niveau k par les liens-suffixes (en vert). d) les noeuds atteints par la remontée précédente voient leur tableau de bits propagé jusqu'à la racine (trait continu). Les noeuds non atteints sont également remontés jusqu'à rencontrer un noeud atteint, les chemins ainsi parcourus (en pointillés) sont disqualifiés. e) l'arbre ainsi modifié est prêt pour une extraction des secondes boîtes de m_1 (en gras).

Pseudo-code

Comme précédemment, cette procédure s'appelle initialement avec le modèle structuré $m = ((m_1 = \varepsilon, m_2 = \varepsilon), (d_{min}, d_{max}))$ et le numéro de bloc en cours d'extension (1 pour débuter). La variable k est la longueur requise pour chaque bloc.

Algorithme 2

ÉpelleModèlesStruct(Modèle m, Bloc i)
Si $i = 2$ **Alors**
  Pour chaque noeud-occurrence v de m **Faire**
    Placer dans $FinsPotentielles$ les fils de v se situant entre les niveaux $2k + d_{min}$ et $2k + d_{max}$
    Pour chaque noeud-occurrence w de $FinsPotentielles$ **Faire**
      Suivre le lien-suffixe rapide vers le noeud-occurrence z du niveau k
      Stocker $Colors_z$ dans L
      Si c'est la première visite de z pour ce modèle m_1 depuis un lien-suffixe rapide **Alors**
        Initialiser $Colors_z$
        Placer z dans la liste $NoeudsAtteints$
      Fin si
      Additionner $Colors_w$ a $Colors_z$
    FinPour
  FinPour
  Faire un parcours préfixe de $\mathcal{A}$ pour mettre à jour les tableaux de bits $Colors$ depuis la racine jusqu'aux noeuds stockés dans $NoeudsAtteints$ (soit $\mathcal{A}'$ l'arbre ainsi modifié).
  $Arbre = \mathcal{A}'$
Sinon
  $Arbre = \mathcal{A}$
FinSi

Pour chaque modèle valide m_i obtenu en descendant simultanément l'arbre des suffixes $Arbre$ et l'arbre virtuel des modèles **à partir de la racine de** $Arbre$ (algorithme B page 94).
Faire
  Si $i = 1$ **Alors**
    ÉpelleModèlesStruct($m = ((m_1, m_2 = \varepsilon), (d_{min}, d_{max}))$, $i + 1$)
  Sinon
    Reporter le modèle $m = ((m_1, m_2), (d_{min}, d_{max}))$ comme valide
  FinSi
FinPour

Si $i = 2$ **Alors**
  Restaurer l'arbre original $\mathcal{A}$ à l'aide du tableau L, par un parcours préfixe de $Arbre$ et une propagation des $Colors$ qu'il contient.
FinSi

Lemme 7

Les deux affirmations suivantes sont vraies :

- *pendant le déroulement de l'algorithme 2, l'arbre $\mathcal{A}'$ construit après l'extraction d'une première boîte m_1 ne contient que les facteurs de longueur k commençant à une distance comprise entre $d_{min}+1$ et $d_{max}+1$ de la dernière position d'une occurrence de m_1 dans les séquences $s_1, \ldots, s_N$;*
- *l'algorithme 2 résout le problème 1.*

Preuve : Observons d'abord qu'à la fin de la ligne 3, *FinsPotentielles* contient tous les noeuds-occurrences associés aux positions finales d'une association « occurrence v de m_1 suivie d'une seconde boîte » respectant la contrainte de distance imposée. À cet instant, ces noeuds-occurrences ont une longueur $2k+i$ (avec $i \in [d_{min}; d_{max}]$) et sont constitués d'une occurrence v de longueur k de m_1 suivie d'un *spacer* de longueur i et d'une occurrence potentielle d'une seconde boîte m_2, de longueur k également. En suivant pour chacun de ces noeuds-occurrences de longueur $2k+i$ le lien-suffixe rapide qui nous ramène au niveau k, nous obtenons le suffixe de longueur k de ce noeud-occurrence, et donc le noeud correspondant à une occurrence potentielle pour m_2. Le fait de suivre les liens-suffixes dépossède ces occurrences de leur contexte : avant, elles sont associées à une occurrence v de m_1 ; après, elles deviennent de simples occurrences de longueur k. Le tableau de bits de ces noeuds du niveau k ne correspond pas au contexte de seconde boîte de ces occurrences ; c'est pourquoi leurs tableaux de bits au niveau $2k+i$ sont propagés au niveau k.

L'ensemble des noeuds atteint par les liens-suffixes rapides est donc l'ensemble des occurrences potentielles d'une seconde boîte m_2 pour m_1. Seuls les noeuds atteints par une remontée via les liens-suffixes rapides sont considérés pour construire $\mathcal{A}'$. Les autres noeuds sont disqualifiés de la participation à sa construction.

L'arbre ainsi créé dispose de toutes les occurrences potentielles d'une seconde boîte pour m_1. On peut donc construire toutes les secondes boîtes m_2 possibles pour m_1 à partir de ces occurrences, et produire tous les modèles structurés de la forme $((m_1, m_2), (d_{min}, d_{max}))$.□

Complexités

Lemme 8

Pour extraire d'un jeu de N séquences de longueur n tous les modèles structurés de la forme $((m_1, m_2), (d_{min}, d_{max}))$, chaque bloc ayant une longueur k et un nombre de substitutions maximal e, la complexité en temps dans le pire des cas de l'algorithme 2 est en $\mathcal{O}(nN^2 + Nn_k\mathcal{V}^2(e,k) + Nn_{2k+d_{max}}\mathcal{V}(e,k))$, et sa complexité en espace est en $\mathcal{O}((2k+d_{max}+N)nN + n_kN)$ sur un alphabet de taille constante.

Preuve : Il nous faut considérer le nombre d'opérations nécessaires pour construire les deux blocs de chaque modèle dans $\mathcal{A}$ ou $\mathcal{A}'$, ainsi que le temps nécessaire à la construction de ces deux arbres et à la restauration de $\mathcal{A}$.

La construction des deux boîtes composant un modèle se fait en temps proportionnel à $2Nn_k\mathcal{V}^2(e,k)$ au plus, puisque chacune des deux extractions est faite dans la partie haute de l'arbre : il y a au plus $2n_k$ noeuds entre la racine de l'arbre et le niveau k, $\mathcal{V}^2(e,k)$

occurrences possibles pour un modèle de deux boîtes de longueur k avec e substitutions dans chaque boîte, et la visite de chaque noeud coûte toujours un temps N.

Le coût des rapatriements des informations contenues entre les niveaux $2k + d_{min}$ et $2k + d_{max}$ (par le biais des liens-suffixes rapides) et de la transformation de $\mathcal{A}$ en $\mathcal{A}'$ (un parcours en profondeur classique) pour tous les modèles m_1 est majoré par :

$$\underbrace{(\sum_{i=d_{min}}^{d_{max}} Nn_{2k+i}\mathcal{V}(e,k))}_{\text{propagations depuis les niveaux } 2k+i} + \underbrace{(2Nn_k\mathcal{V}(e,k))}_{\text{transformation de } \mathcal{A} \text{ en } \mathcal{A}'} \leq 2Nn_{2k+d_{max}}\mathcal{V}(e,k)$$

La restauration de $\mathcal{A}$ est aussi un parcours en profondeur des k niveaux de $\mathcal{A}'$ et coûte donc $2Nn_k\mathcal{V}(e,k)$ opérations. La construction de l'arbre initial coûte toujours un temps en $\mathcal{O}(nN^2)$. La complexité totale en temps pour l'algorithme 2 est donc en $\mathcal{O}(nN^2 + 2Nn_k\mathcal{V}^2(e,k) + 2Nn_{2k+d_{max}}\mathcal{V}(e,k) + Nn_k\mathcal{V}(e,k))$ ce qui est équivalent à $\mathcal{O}(nN^2 + Nn_k\mathcal{V}^2(e,k) + Nn_{2k+d_{max}}\mathcal{V}(e,k))$.

La complexité en espace est évidemment plus élevée que pour l'algorithme 1. Il faut en effet ajouter aux coûts de l'arbre et des occurrences à stocker durant l'extraction celui du tableau L servant à stocker temporairement les tableaux de bits des noeuds du niveau k : $\mathcal{O}(n_kN)$, où $n_k \leq nN$. On obtient donc une complexité en espace en $\mathcal{O}((2k + d_{max} + N)nN + n_kN)$.$\square$

Si cet algorithme apporte un surcoût en espace par rapport à l'algorithme 1, sa complexité en temps est en revanche meilleure. Il permet en théorie de réduire le coût excessif d'un long saut ou de plusieurs sauts successifs, qui génèrent un nombre important d'occurrences dans l'algorithme 1. Son implantation est cependant plus délicate du fait des modifications importantes à apporter à l'arbre pendant le déroulement de l'algorithme.

Notons que, dans cet algorithme comme dans le précédent, le coût du cardinal de l'alphabet intervient de la même manière que dans l'algorithme d'extraction de modèles simples : un facteur $\log \Sigma$ en temps et un facteur $|\Sigma|$ en espace lors de la construction de l'arbre, et une influence exponentielle sur la taille du voisinage $\mathcal{V}(e,k)$ d'un modèle.

5.2.2 Extensions aux modèles à p boîtes (Problème 2)

Dans cette section, nous étendons les deux meilleurs algorithmes du problème 1 au problème 2, afin de généraliser l'extraction de deux boîtes au cas de p boîtes. La figure 5.6 montre les caractéristiques des modèles de la forme $((m_1, m_2, \ldots, m_p), (d_{min}, d_{max}))$ que nous souhaitons maintenant extraire.

5.2.2.1 Algorithme 1

Description

L'extension de l'algorithme 1 au traitement de $p > 2$ boîtes est simple : il suffit de répéter

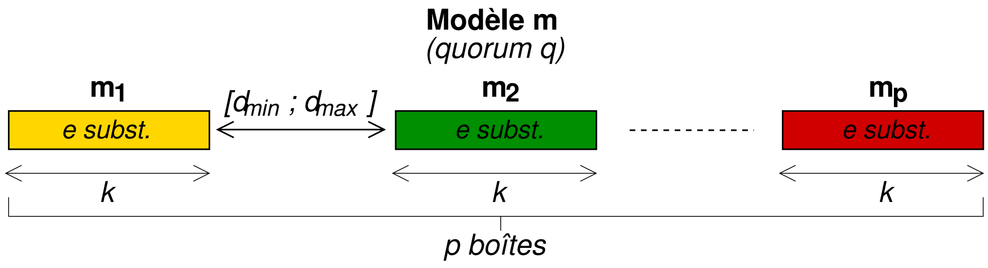

FIG. 5.6 – Caractéristiques des modèles à extraire dans le problème 2.

le principe de saut dans l'arbre autant de fois que nécessaire. Après l'extraction de chaque boîte $m_i, i < p$, on étend ses occurrences de d_{min} à d_{max} et on procède à l'extraction de m_{i+1} avec le jeu d'occurrences obtenu.

Pseudo-code

L'algorithme 1 étendu au problème 2 s'appelle de la même façon qu'auparavant. Très peu de modifications interviennent.

```
ÉpelleModèlesStruct(Modèle m, Bloc i)
Si i > 1 Alors
  Pour chaque noeud-occurrence v de m Faire
    Placer dans DebutsPotentiels les fils de v se situant
      entre les niveaux (i − 1)(k + d_min) et (i − 1)(k + d_max)
  FinPour
Sinon
  Placer v (la racine de l'arbre) dans DebutsPotentiels
FinSi

Pour chaque modèle valide m_i obtenu en descendant simultanément
  l'arbre des suffixes A et l'arbre virtuel des modèles
  à partir des noeuds-occurrences de DebutsPotentiels
  (algorithme B page 94) Faire
  Si i < p Alors
    ÉpelleModèlesStruct(m = ((m_1, ..., m_i = ε, ...), (d_min, d_max)), i + 1)
  Sinon
    Reporter le modèle m = ((m_1, ..., m_p), (d_min, d_max)) comme valide
  FinSi
FinPour
```

Complexités

Lemme 9
Pour extraire d'un jeu de N séquences de longueur n tous les modèles structurés de la forme $((m_1, \ldots, m_p), (d_{min}, d_{max}))$, chaque bloc ayant une longueur k et un nombre de substitutions maximal e, la complexité en temps dans le pire des cas de l'algorithme 1 est en $\mathcal{O}(nN^2 + Nn_{pk+(p-1)d_{max}}\mathcal{V}^p(e,k))$, et sa complexité en espace est en $\mathcal{O}((p(k+d_{max})+N)nN)$ sur un alphabet de taille constante.

Preuve : Pour la complexité en temps, suivant le même raisonnement que dans le cas de deux boîtes, on multiplie le nombre de noeuds au dernier niveau $pk + (p-1)d_{max}$ par le nombre de modèles à p boîtes pouvant atteindre un noeud de ce niveau $\mathcal{V}^p(e,k))$ et par le coût de la visite d'un noeud (N). On obtient une complexité en temps de $\mathcal{O}(Nn_{pk+(p-1)d_{max}}\mathcal{V}^p(e,k))$.

Le nombre d'occurrences à stocker pendant le déroulement de l'algorithme est maintenant majoré par $(pk + (p-1)d_{max})nN$. La complexité en espace est donc en $\mathcal{O}((pk + (p-1)d_{max} + N)nN)$ ce qui équivaut à $\mathcal{O}((p(k + d_{max} + N)nN)$.$\square$

5.2.2.2 Algorithme 2

L'extension de l'algorithme 2 à la gestion de p boîtes est plus complexe. Il faut en effet gérer i instances différentes de l'arbre $\mathcal{A}$ pour extraire la $i^{\text{ème}}$ boîte.

Description

Les extractions de boîtes sont réalisées de la manière habituelle. Ce sont les transitions entre ces extractions qui doivent être modifiées. Les opérations de modifications de l'arbre entre l'extraction de $m_{i<p}$ et m_{i+1} diffèrent sur deux points. En premier lieu, il faut maintenant stocker i tableaux L afin de pouvoir remonter la récursion correctement et rétablir les arbres précédents.

Sur le second point prenons l'exemple de la transition entre l'extraction d'une seconde et d'une troisième boîte. Pour un m_2 donné, on atteint le niveau k avec des occurrences pour $((m_1, m_2), (d_{min}, d_{max}))$. Pour procéder à l'extraction d'une troisième boîte, <u>on ne peut pas étendre ces occurrences plus avant</u>. En effet, l'arbre $\mathcal{A}'$ dans lequel nous travaillons n'a été mis à jour que jusqu'au niveau k. Au-delà, on risque d'emprunter des chemins correspondants à des occurrences de m_2 seul (et non plus du couple (m_1, m_2)), et donc de grossir la couverture du modèle en cours avec des occurrences qui ne devraient pas lui appartenir.

Pour construire un nouvel arbre $\mathcal{A}''$ permettant l'extraction d'une troisième boîte, il nous faut étendre les occurrences que nous avions quittées aux niveaux $2k+d_{min}$ à $2k+d_{max}$ vers les niveaux $3k + 2d_{min}$ à $3k + 2d_{max}$ où se trouvent les noeuds correspondant aux positions finales d'une association « occurence de m_1 suivie d'une occurrence de m_2 suivie d'une occurrence de m_3 » respectant les contraintes de distance imposées. Une fois ces noeuds atteints, on remontera au niveau k par les liens-suffixes rapides pour générer le nouvel arbre, comme auparavant.

La seule modification qui intervient ici est donc qu'il faut mémoriser au niveau k les noeuds-occurrences des fins potentielles se situant en bas de l'arbre et qui ont permis de les atteindre par le biais des liens-suffixes. Ces noeuds sont stockés dans un tableau de listes de pointeurs, chaque noeud du niveau k étant associé à une cellule de ce tableau.

Pseudo-code

$Ptr_{i>0}$ indique le $i^{\text{ème}}$ tableau de listes de pointeurs des noeuds du niveau k vers les noeuds des niveaux $(i+1)k+id_{min}$ à $(i+1)k+id_{max}$ d'où ils proviennent. La variable L_i indique le $i^{\text{ème}}$ tableau de stockage des tableaux $Colors$ du niveau k permettant de restaurer l'arbre précédent. Soit $\mathcal{A}^{(1)} = \mathcal{A}$.

ÉpelleModèlesStruct(Modèle m, Bloc i)
Si $i > 1$ **Alors**
  Pour chaque noeud-occurrence $Ptr_{i-1}(v)$ de m **Faire**
    Placer dans $FinsPotentielles$ les fils de v se situant
      entre les niveaux $ik + (i-1)d_{min}$ et $ik + (i-1)d_{max}$
    Pour chaque noeud-occurrence w de $FinsPotentielles$ **Faire**
      Suivre le lien-suffixe rapide vers le noeud-occurrence z du niveau k
      Stocker $Colors_z$ dans L_{i-1}
      Si c'est la première visite de z pour ce modèle m_{i-1}
        depuis un lien-suffixe rapide **Alors**
        Initialiser $Colors_z$ et la liste $Ptr_i(z)$
        Placer z dans la liste $NoeudsAtteints$
      Fin si
      Additionner $Colors_w$ à $Colors_z$
      Ajouter un pointeur vers le noeud w dans la liste $Ptr_i(z)$
    FinPour
  FinPour
  Faire un parcours préfixe de $\mathcal{A}$ pour mettre à jour les tableaux
  de bits $Colors$ depuis la racine jusqu'aux noeuds stockés
  dans $NoeudsAtteints$ (soit $\mathcal{A}^{(i)}$ l'arbre ainsi modifié)
    $Arbre = \mathcal{A}^{(i)}$
Sinon
  $Arbre = \mathcal{A}$
FinSi

Pour chaque modèle valide m_i obtenu en descendant simultanément l'arbre
  des suffixes $Arbre$ et l'arbre virtuel des modèles **à partir de la racine**
  de $Arbre$ (algorithme B page 94) **Faire**
  Si $i = 1$ **Alors**
    Pour chaque occurrence v de m_1, $Ptr_1(v) = v$
  FinSi
  Si $i < p$ **Alors**
    EpelleModelesStruct($m = ((m_1, m_2 = \varepsilon, \ldots), (d_{min}, d_{max}))$, $i + 1$)
  Sinon
    Reporter le modèle $m = ((m_1, \ldots, m_p), (d_{min}, d_{max}))$ comme valide
  FinSi
FinPour
Si $i > 1$
  Restaurer l'arbre précédent $\mathcal{A}^{(i-1)}$ à l'aide du tableau L_{i-1},
    par un parcours préfixe de $Arbre$ et une propagation de $Colors$
FinSi

Complexités

Lemme 10
Pour extraire d'un jeu de N séquences de longueur n tous les modèles structurés de la forme $((m_1,\ldots,m_p),(d_{min},d_{max}))$, chaque bloc ayant une longueur k et un nombre de substitutions maximal e, la complexité en temps dans le pire des cas de l'algorithme 2 est en $\mathcal{O}(nN^2 + Nn_k\mathcal{V}^p(e,k) + Nn_{pk+(p-1)d_{max}}\mathcal{V}^{p-1}(e,k))$, et sa complexité en espace est en $\mathcal{O}((p(k+d_{max})+N)nN + n_{pk+(p-1)d_{max}}N(p-1))$ sur un alphabet de taille constante.

Preuve : Suivant le même raisonnement que dans le cas de deux boîtes, le nombre d'opérations nécessaires à la construction des modèles à p boîtes est majoré par $\mathcal{O}(Nn_k\mathcal{V}^p(e,k))$.

Le nombre total d'opérations nécessaires à la transformation de tous les arbres $\mathcal{A}^{(i<p)}$ en $\mathcal{A}^{(i+1)}$ (entre l'extraction des m_i et celle des m_{i+1}) est majoré par :

$$(\sum_{j=d_{min}}^{d_{max}} Nn_{(i+1)k+ij}\mathcal{V}^i(e,k)) + (Nn_k\mathcal{V}(e,k)) \leq 2Nn_{(i+1)k+id_{max}}\mathcal{V}^i(e,k)$$

Si on étend cette formule aux $(p-1)$ transitions à faire entre les p boîtes, on obtient une complexité en $\mathcal{O}(pNn_{pk+(p-1)d_{max}}\mathcal{V}^{p-1}(e,k))$ pour cette étape, et une complexité totale en temps en $\mathcal{O}(nN^2 + Nn_k\mathcal{V}^p(e,k) + pNn_{pk+(p-1)d_{max}}\mathcal{V}^{(p-1)}(e,k))$.

En espace, il faut ajouter au coût de l'arbre celui des $p-1$ tableaux L et des $p-1$ tableaux de pointeurs Ptr $(\mathcal{O}(n_{pk+(p-1)d_{max}}N(p-1)))$, ainsi que le coût des occurrences à stocker durant le déroulement de l'algorithme $((p(k+d_{max}))nN$. La complexité en espace est donc en $\mathcal{O}((p(k+d_{max})+N)nN + n_{pk+(p-1)d_{max}}N(p-1))$.□

Toujours moyennant un surcoût en espace, l'algorithme 2 apporte une complexité meilleure en temps que celle de l'algorithme 1.

5.2.3 Extensions à des modèles à intervalles de saut indéterminés (Problème 3)

Le problème 3 introduit la notion d'intervalle de saut indéterminé entre deux boîtes. On ne travaille plus avec un intervalle de saut précis, on sait seulement que l'intervalle de saut entre les deux boîtes a une largeur 2δ, et qu'il se situe lui-même dans l'intervalle $[d_{min};d_{max}]$. On cherche à extraire les modèles de la forme $((m_1,m_2),(d_{min},d_{max},\delta))$, dont les caractéristiques sont décrites figure 5.7.

5.2.3.1 Algorithme 1

La modification à faire dans cet algorithme (par rapport à la version du problème 1) consiste juste à distinguer les occurrences d'après-saut en fonction de la longueur du saut effectué. Ainsi, à chaque étape de l'extraction de la seconde boîte, au moment de calculer la validité du modèle, on fait $\Delta' = d_{max} - d_{min} - 2\delta + 1$ (le nombre d'intervalles valides)

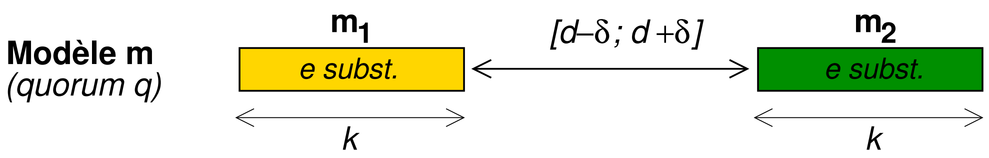

FIG. 5.7 – Caractéristiques des modèles à extraire dans le problème 3.
Avec $d \in [d_{min} + \delta ; d_{max} - \delta]$.

calculs de couverture avec le lot d'occurrences dont la longueur de saut appartient à chacun des intervalles possibles.

Il suffit donc de trier les occurrences obtenues après chaque saut entre deux boîtes selon la longueur du saut effectué dans l'occurrence, et de modifier légèrement le processus de validation d'une boîte afin de transmettre les intervalles sur lesquels le quorum a été atteint.

Lemme 11
Pour extraire d'un jeu de N séquences de longueur n tous les modèles structurés de la forme $((m_1, m_2), (d_{min}, d_{max}, \delta))$, chaque bloc ayant une longueur k et un nombre de substitutions maximal e, la complexité en temps dans le pire des cas de l'algorithme 1 est en $\mathcal{O}(nN^2 + \delta N n_{2k+d_{max}} \mathcal{V}^2(e,k))$, et sa complexité en espace est en $\mathcal{O}((2k + d_{max} + N)nN)$, sur un alphabet de taille constante.

Preuve : La seule modification à apporter consiste à stocker, pour chaque occurrence, les sauts qu'elle a suivis afin de pouvoir permettre un calcul de quorum séparé. On ajoute donc un facteur qui est constant au coût $(2k + d_{max})nN$ de stockage des occurrences durant le déroulement de l'algorithme. Lors de chaque visite d'un noeud, celui-ci peut-être impliqué dans 2δ+1 calculs de quorum, puisqu'il peut appartenir a autant d'intervalles. En suivant les raisonnements précédents, on obtient un coût en temps en $\mathcal{O}(nN^2 + \delta N n_{2k+d_{max}} \mathcal{V}^2(e,k))$.□

5.2.3.2 Algorithme 2

L'adaptation de cet algorithme est encore une fois plus complexe, puisqu'il faut faire apparaître dans l'arbre $\mathcal{A}'$ les informations concernant la longueur du saut emprunté.

En conservant la philosophie de cet algorithme, il nous faut donc modifier les k premiers niveaux de l'arbre $\mathcal{A}'$ de façon à ce qu'ils contiennent les informations dont nous avons besoin : la couverture de chaque occurrence de la seconde boîte en fonction de la longueur du saut qui a été effectué pour l'atteindre.

Pour ce faire, nous remplaçons les tableaux $Colors_n$ (de taille N) de chaque noeud n de $\mathcal{A}'$ par des tableaux $MColors_n[l]$ ($l \in [d_{min} + \delta ; d_{max} - \delta]$) à deux dimensions, où à chaque centre d'intervalle de saut possible (d dans la figure 5.7) est associé un tableau de bits classique de taille N.

Ainsi, lors de la remontée dans $\mathcal{A}$ depuis les noeuds des niveaux $2k + d_{min}$ à $2k + d_{max}$

via les liens-suffixe rapides vers le niveau k, on copie comme auparavant le tableau de bits du noeud w d'origine dans le noeud z atteint, mais cette fois dans les cellules du tableau $MColors_z$ correspondant aux intervalles qui sont concernés par cette occurrence. Plus précisément, pour un noeud-occurrence w au niveau $2k+i$, on remonte au noeud-occurrence z associé et on positionne $MColors_z$ de la façon suivante :

$$\forall i \in [\min(d_{max} + \delta, i - \delta); \max(d_{max} - \delta, i + \delta)],$$

$$MColors_z[i] = MColors_z[i] \cup Colors_w$$

Ensuite, ces tableaux seront propagés dans $\mathcal{A}'$, comme auparavant par addition, et ce pour chaque intervalle de saut séparément.

Au moment de déterminer la validité d'un modèle (pendant l'extraction de la seconde boîte uniquement), on calculera $d_{max} - d_{min} - 2\delta + 1$ couvertures distinctes. Seuls les intervalles ayant obtenu le quorum requis seront étendus plus avant (on distingue, pour le modèle en cours d'extraction, les intervalles qui restent valides et ceux qui ne le sont plus).

Lemme 12
Pour extraire d'un jeu de N séquences de longueur n tous les modèles structurés de la forme $((m_1, m_2), (d_{min}, d_{max}, \delta))$, chaque bloc ayant une longueur k et un nombre de substitutions maximal e, la complexité en temps dans le pire des cas de l'algorithme 2 est en $\mathcal{O}(N\Delta' n_k \mathcal{V}^2(e,k) + N\Delta' n_{2k+d_{max}} \mathcal{V}(e,k))$ où $\Delta' = d_{max} - d_{min} - 2\delta + 1$, et sa complexité en espace est en $\mathcal{O}((2k + d_{max} + N)nN + n_k\Delta' N)$, sur un alphabet de taille constante.

Preuve : En reprenant le calcul de complexité en temps du problème 1, on constate qu'il nous faut introduire le facteur Δ' dans certains termes :
- dans la propagation des noeuds du niveau k jusqu'à la racine, puisqu'on doit maintenant propager Δ' tableaux de taille N pour chaque noeud ;
- dans l'extraction de la seconde boîte, puisque la visite d'un noeud coûte alors un temps $\mathcal{O}(\Delta N)$.

On obtient donc une complexité en temps en $\mathcal{O}(N\Delta' n_k \mathcal{V}^2(e,k) + N\Delta' n_{2k+d_{max}} \mathcal{V}(e,k))$.

Le tableau L de stockage des tableaux $MColors$ occupe maintenant un espace en $\mathcal{O}(n_k\Delta' N)$, d'où une complexité en espace en $\mathcal{O}((2k + d_{max} + N)nN + n_k\Delta' N)$.$\square$

5.2.4 Algorithmes complets (Problème 4)

Dans cette section, nous fusionnons finalement naturellement les algorithmes des problèmes 2 et 3 pour résoudre le problème 4. Nous résolvons ainsi le problème le plus générique en proposant deux algorithmes d'extraction de modèles structurés de la forme $((m_1, \ldots, m_p), (d_{min}, d_{max}, \delta))$. Les caractéristiques de ces modèles sont décrits figure 5.8.

5.2.4.1 Algorithme 1

En réunissant les particularités de l'algorithme 1 du problème 2 (simple extension du principe de saut dans l'arbre) et du problème 3 (mémorisation des distances de saut pour

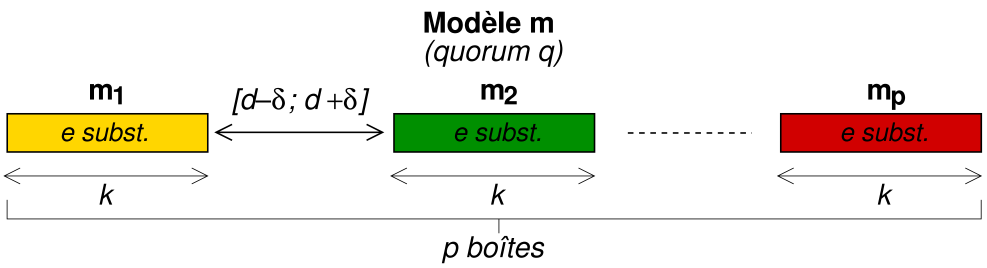

FIG. 5.8 – Caractéristiques des modèles à extraire dans le problème 4. Avec $d \in [d_{min} + \delta; d_{max} - \delta]$.

chaque occurrence), on déduit facilement une instance de cet algorithme pour le problème 4 . Il suffit de stocker non plus la longueur d'un, mais de tous les sauts réalisés pour une occurrence donnée. Hormis un surcoût en espace de $p - 1$ sauts à stocker par occurrence, apparaît également un surcoût en temps lors de l'évaluation des quorums : chaque occurrence (de la dernière boîte) peut en effet appartenir à $(2\delta + 1)^p$ intervalles différents. On obtient donc une complexité en temps en $\mathcal{O}(nN^2 + \delta^p N n_{pk+(p-1)d_{max}} \mathcal{V}^p(e,k))$ et une complexité en espace en $\mathcal{O}((p^2(k + d_{max}) + N)nN)$.

Ces résultats dépendent néanmoins beaucoup de l'implantation choisie pour effectuer les calculs de quorum par intervalles. La version que nous avons implantée ne suit pas exactement les méthodes décrites ici : elle évite les calculs de quorums distincts à chaque étape en lançant, après chaque saut, $\Delta' = d_{max} - d_{min} - 2\delta + 1$ récursions distinctes pour chaque intervalle valide. Elle est donc plus coûteuse en temps mais moins en espace puisqu'on n'a pas à stocker tous les sauts réalisés par chaque occurrence.

5.2.4.2 Algorithme 2

L'extension de cet algorithme depuis la version du problème 3 consiste à effectuer les mêmes modifications que pour le passage du problème 1 au problème 2. Il suffit de se doter d'un tableau L permettant de reconstruire $p-1$ arbres de hauteur k dont les noeuds possèdent des tableaux de bits de dimension $\Delta' N$. La complexité en espace passe donc à $\mathcal{O}((p(k + d_{max}) + N)nN + n_{pk+(p-1)d_{max}} \Delta N(p-1))$.

La complexité en temps subit les mêmes altérations :
$\mathcal{O}(nN^2 + N\Delta'(p-1)n_k\mathcal{V}^p(e,k) + N\Delta' n_{pk+(p-1)d_{max}}\mathcal{V}^{p-1}(e,k))$.

5.2.5 Mises au point

Nous avions convenu d'extraire des modèles aux caractéristiques simplifiées, afin de pouvoir nous focaliser sur les aspects algorithmiques du problème. Nous pouvons maintenant étendre les algorithmes obtenus à des modèles possédant des tailles de boîtes variables, des longueurs de sauts différentes selon les intervalles concernés, un nombre de substitutions global ou d'autres critères.

5.2.5.1 Critères quelconques des boîtes et sauts

Pour simplifier, nous avions décidé de résoudre les quatre problèmes posés pour des modèles ayant des boîtes de longueur k, des sauts dans l'intervalle $[d_{min}; d_{max}]$ et éventuellement des deltas δ, ces caractéristiques étant identiques pour chaque boîte ou modèle concerné.

Dans le modèle le plus général possible, chaque boîte a une taille ou une fourchette de taille propre. Les intervalles de saut et les deltas sont également distincts entre chaque couple de boîtes successives.

Concernant l'algorithme 1, ces extensions sont triviales, on descend et on saute dans l'arbre en suivant scrupuleusement les caractéristiques du modèle à extraire. Aucune modification n'est à apporter dans l'algorithme si ce n'est le traitement de ces nouveaux critères. Les complexités sont modifiées en conséquence.

Pour l'algorithme 2, il y a par contre quelques modifications à apporter. En effet, les extractions sont réalisées dans un arbre construit dynamiquement. Or cet arbre doit avoir une hauteur correspondant à la longueur maximum de la boîte à extraire. Les informations collectées dans le bas de l'arbre doivent donc être ramenées au niveau correspondant à cette longueur maximum. Toutes les complexités sont donc modifiées en fonction des longueurs maximales des boîtes. Les tableaux servant à restaurer l'arbre précédent ou à atteindre les noeuds du bas de l'arbre ont également une taille qui dépend de ces longueurs de boîtes. La façon dont les liens-suffixes rapides sont construits dépend également de ces longueurs de boîtes. Enfin, dans le cas où on l'utilise, le δ influe également sur le nombre de tableaux de bits à propager et à stocker.

Notons que l'algorithme 2 nécessite de fixer une longueur maximum pour chaque boîte, ceci afin de savoir sur quelle hauteur modifier l'arbre. L'utilisation d'un paramètre de longueur $k_{max} = \infty$ pour une des boîtes n'est donc pas possible.

5.2.5.2 Critères globaux

Les stratégies employées dans les deux algorithmes que nous avons développés permettent de prendre en compte facilement de nombreuses caractéristiques globales, créant ainsi certaines corrélations entre les différentes boîtes d'un modèle structuré. On peut par exemple modifier l'algorithme pour qu'un nombre de substitutions global soit pris en compte, ou bien que la composition en certains symboles de l'alphabet soit limitée.

On peut également aisément imposer que deux boîtes soient des palindromes, ou bien que des boîtes successives soient identiques (ou proches) comme on l'observe dans les sites de fixation de certains eucaryotes, voire chez des bactéries.

5.3 Extensions maximales

Les travaux décrits dans cette section ont été réalisés en collaboration avec Nadia Pisanti, du Dipartimento di Informatica de l'Università di Pisa (Italie), durant l'été/automne 2001, et sont toujours en cours.

L'algorithme original de Marie-France Sagot, et donc ceux que nous avons développés qui en découlent, ont pour but d'éviter la visite de la totalité de l'espace des solutions en se basant sur des propriétés que des mots ayant une relation entre eux partagent. Plus précisément, cet algorithme évite de visiter les modèles dont un préfixe ne respecterait pas le quorum requis (grâce au lemme 2, page 86). Nous proposons ici une amélioration basée sur le constat suivant : lors du déroulement de l'algorithme, nous recueillons des informations sur les extensions des modèles visités qui pourraient être exploitées durant les étapes ultérieures du déroulement de cet algorithme.

Les travaux exposés dans cette section sont à ce jour au stade théorique. Nous avons réalisé quelques expériences sur des jeux de données, visant à mesurer la pertinence de cette approche et les gains potentiels. Il nous reste cependant à intégrer cette optimisation dans SMILE afin d'évaluer son coût dans les conditions réelles d'une extraction. Les calculs de complexité dans le pire des cas n'apportent pas d'améliorations aux complexités précédentes ; il faudrait ici (comme dans toutes les preuves précédentes, qui font des majorations très larges) s'attacher à déterminer des complexités en moyenne pour obtenir des bornes plus réalistes.

5.3.1 Prévoir à l'avance l'invalidation d'un modèle

L'idée que nous avons utilisée pour optimiser nos algorithmes consiste à prévoir à l'avance qu'un modèle n'atteindra jamais la taille requise.

Considérons que nous procédons à une extraction des modèles de longueur k_{min} à k_{max} avec un quorum q. Lors du parcours en profondeur de l'arbre virtuel de tous les modèles, on s'aperçoit qu'aucun modèle commençant par $w \in \Sigma^{j<k_{min}}$ n'a atteint la longueur minimum k_{min} requise. Nous dirons qu'aucun modèle ayant w pour préfixe n'a pu être étendu de plus de $MaxExt(w)$ symboles.

Définition 12 (Extension maximale)
Soit $w \in \Sigma^*$ un modèle de quorum q. On note $MaxExt(w)$ et on désigne par extension maximale *de w le nombre maximal de symboles dont on peut étendre w sans qu'il ne soit invalidé (quorum q non respecté).*

L'idée que nous souhaitons exploiter est de profiter de l'information $MaxExt(w)$ que l'on aurait obtenue auparavant, durant le parcours des modèles, pour devancer l'éventuelle invalidation d'un modèle x ayant w pour suffixe.

Lemme 13
Dans le contexte d'une extraction de modèles de longueur k_{min} à k_{max}, étant donné un modèle $w \in \Sigma^*$, si il existe un modèle $x \in \Sigma^+$ dont w est un des suffixes, et que $|x| + MaxExt(w) < k_{min}$, il est inutile de poursuivre plus avant l'extension de x, qui n'atteindra pas la longueur minimum k_{min} requise.

Preuve : Par définition, le modèle w ne peut être étendu de plus de $MaxExt(w)$ sans être invalidé, il n'y a donc pas de modèles de longueur supérieure à $|w| + MaxExt(w)$ et ayant w pour préfixe qui soit valide.

Or, x ayant w pour suffixe, il a au plus les mêmes occurrences que w. Par conséquent, son extension se déroulera au mieux de la même façon que celle de w, avec au plus $MaxExt(w)$ symboles. Si cette extension maximale prévue ne suffit pas à atteindre la longueur requise, alors il est inutile de poursuivre l'extension de x.□

Sans cette optimisation, notons que l'algorithme réalise pour x toutes ou partie des extensions déjà réalisées pour w avant de se rendre compte que x ne donnera pas naissance à un modèle valide de longueur minimum k_{min}. Ce sont ces extensions, inutiles parce que l'on sait qu'elles ne porteront pas assez loin, que nous souhaitons économiser. La figure 5.9 illustre un cas où cette optimisation est possible.

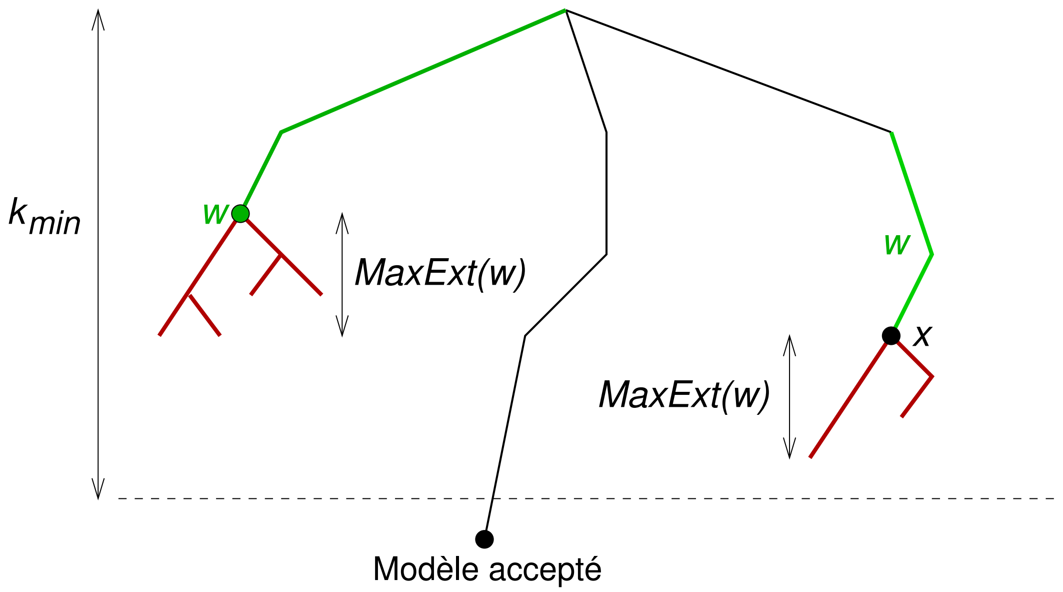

FIG. 5.9 – Optimisation par les extensions maximales.
L'arbre des modèles représenté ici est parcouru de façon préfixe (de gauche à droite) lors d'une extraction. Le modèle w recueille un $MaxExt(w)$ durant cette descente. Lors de la visite du modèle x, on interrompt la descente en constatant que x a w pour suffixe et que $|x| + MaxExt(w) < k_{min}$.

Notre optimisation se base donc sur l'utilisation des extensions maximales des suffixes du modèle en cours afin d'essayer d'interrompre prématurément son extension. Dans les sections suivantes, nous tâchons de répondre aux questions suivantes :

- comment calculer ces extensions maximales de façon efficace ?
- comment les stocker de façon efficace ?
- comment en profiter de façon efficace ?

Comme à notre habitude, nous considérons d'abord une instance simplifiée du problème que nous nous posons. Nous cherchons donc à inclure cette optimisation dans l'algorithme original d'extraction de modèles simples. Nous l'étendrons ensuite aux algorithmes d'extraction de modèles structurés que nous avons développés.

5.3.2 Calculer les extensions maximales

L'algorithme de Marie-France Sagot réalise un parcours en profondeur de l'arbre virtuel des modèles (que nous noterons $\mathcal{M}$), et parcourt parallèlement l'arbre des suffixes des séquences (noté $\mathcal{A}$) pour y rechercher les occurrences du modèle en cours de visite.

Soient $m \in \Sigma^*$ et $\alpha \in \Sigma$. L'extension du modèle $m\alpha$ cesse quand l'une de ces conditions est vérifiée (section 4.2.3, page 87) :

- il a atteint la longueur maximum k_{max} requise ;
- il ne respecte plus le quorum q.

Dans le premier cas, la véritable extension maximale de $m\alpha$ ne nous intéresse pas puisque ce dernier a atteint la longueur maximale requise et ne sera plus étendu. On considèrera donc qu'on ne connaît pas son extension maximale, que l'on positionnera à $MaxExt(m\alpha) = \infty$.

Dans le second cas, $m\alpha$ ne respecte plus le quorum alors que ses préfixes le respectaient. À cette étape, on peut déjà déduire immédiatement que $MaxExt(m\alpha) = 0$. Mais la remontée de récursion qui se produit après cette disqualification de $m\alpha$ nous permet aussi de calculer d'autres extensions maximales. En effet, après α c'est l'extension de m vers le symbole qui suit dans l'alphabet (disons β) qui est réalisée. Par hypothèse de récurrence, considérons que cette nouvelle extension produit toujours un $MaxExt(m\beta)$, et qu'il en va de même pour tous les symboles de Σ avec lesquels on va étendre m. Une fois le dernier symbole de l'alphabet utilisé pour étendre m, la descente dans le sous-arbre (virtuel) enraciné au noeud correspondant à m prend fin. On peut alors déduire $MaxExt(m)$ par la formule qui suit :

$$MaxExt(m) = \max_{\alpha \in \Sigma}(MaxExt(m\alpha)) + 1$$

En effet, l'extension maximale d'un modèle vaut la plus grande des extensions de ce modèle étendu d'un symbole, augmentée de 1.

Ainsi, par récurrence, tout modèle visité lors du parcours de $\mathcal{M}$ et dont le sous-arbre a été intégralement visité recueille une extension maximale. De proche en proche et en dernier lieu, la racine r de l'arbre $\mathcal{M}$ recevra dans $MaxExt(r)$ la longueur du plus grand modèle valide visité (ou éventuellement ∞ selon la façon dont se sera déroulée l'extraction).

5.3.3 Stocker au mieux les extensions maximales

Nous venons de montrer que pour tous les modèles valides et tous les premiers modèles invalidés, on peut calculer une extension maximale pendant le déroulement normal de l'algorithme. Pour stocker ces informations, il nous faudrait construire un autre arbre, réel celui-ci, que l'on remplirait au fur et à mesure du parcours de $\mathcal{M}$ avec tous les modèles valides rencontrés.

5.3.3.1 L'arbre des suffixes et les noeuds fantômes

Pour éviter de stocker les $MaxExt$ dans une nouvelle structure qui pourrait s'avérer coûteuse, nous décidons d'utiliser la structure existante : l'arbre des suffixes (ou des facteurs) $\mathcal{A}$. Cependant, cette structure ne permet pas toujours, telle que nous l'utilisons jusqu'alors, de stocker ces informations.

Posons ici deux hypothèses, dont nous montrerons d'abord facilement qu'elles sont fausses, puis dont nous montrerons ensuite qu'elles peuvent être fréquemment vraies sous certaines conditions :

1. Tous les modèles valides ont une occurrence exacte dans les séquences.
2. Tous les mots des séquences ont un noeud-occurrence dans $\mathcal{A}$.

L'hypothèse 1 est fausse car, comme expliqué quand nous les avons définis, les modèles sont des objets extérieurs à la séquence. Un modèle n'apparaît pas nécessairement de façon exacte dans les séquences. Cependant, un modèle de longueur faible et bien conservé a plus de chances de posséder une occurrence exacte qu'un modèle long et très dégénéré. Si on ne calcule que les extensions maximales des modèles courts, avec les taux de conservation raisonnables utilisés pour l'inférence de sites de fixation, on peut penser qu'une grande partie de ces modèles aura au moins une occurrence exacte.

L'hypothèse 2 est fausse car nous utilisons une structure d'arbre des suffixes compacté. Par conséquent, certaines occurrences des séquences correspondent à des pseudo-noeuds et non à des noeuds. Malgré cela, pour un jeu de séquences donné, plus on est proche de la racine et moins on a de chances de rencontrer un pseudo-noeud. Cela est dû au fait que plus un facteur est court, plus on a de chances de le voir suivi de plusieurs symboles différents le long des séquences.

On observe généralement (voir les résultats obtenus dans [Allali, 2000]) que l'arbre compact des suffixes d'une séquence de longueur n est complet et non-compacté sur ses $\log_{|\Sigma|}(n)$ premiers niveaux. Au dessus de cette limite, on peut supposer que l'arbre des suffixes $\mathcal{A}$ ressemble fortement à l'arbre virtuel des modèles $\mathcal{M}$.

Sur un alphabet à quatre symboles et pour une séquence de 100Kb (kilo-bases), cette limite vaut plus de 8. Pour une séquence de 1Mb (méga-bases) elle vaut près de 10. Cela signifie que, pour les quantités de données sur lesquelles nous travaillons usuellement, et sur l'alphabet de l'ADN, l'arbre des suffixes est en général complet au-dessus du huitième niveau.

Nous souhaitons stocker les informations d'extension maximale d'un modèle m dans le noeud de $\mathcal{A}$ qui épelle m, celui de l'occurrence exacte de m. Les deux hypothèses précédentes, bien que fausses dans l'absolu, pourraient être fréquemment validées à condition de se restreindre au calcul des extensions maximales des modèles courts. Tablons donc sur le fait que nous trouverons souvent un noeud qui correspondra à m dans $\mathcal{A}$ aux niveaux de profondeur inférieure ou égale à $\log_{|\Sigma|}(n)$.

Dans le cas où ce noeud n'existe pas ou bien s'il s'agit d'un pseudo-noeud, il nous faudra l'ajouter dans l'arbre $\mathcal{A}$ (en prenant bien soin de le distinguer des autres noeuds de l'arbre afin qu'il ne soit pas confondu avec une occurrence lors de l'extraction). Nous qualifierons un tel noeud de *noeud fantôme*.

5.3.3.2 Zone d'utilisation des extensions maximales

Nous nous interrogeons maintenant sur la longueur maximale utile des modèles pour lesquels nous devons calculer et stocker l'information $MaxExt$. Le lemme 13 permet d'interrompre l'extension de modèles dont un des suffixes aurait un $MaxExt$ qui le permettrait. Le calcul des extensions maximales des modèles de longueur $k_{min} - 1$ est inutile puisque les modèles qui pourraient en profiter auront déjà atteint la longueur k_{min} et ne pourront donc plus être interrompus dans leur extension par le lemme 13.

Il est donc inutile de stocker les extensions maximales au-delà de la profondeur $k_{min} - 2$ de l'arbre. Cette limitation est la bienvenue puisqu'elle nous oblige à nous restreindre à une zone un peu plus haute de l'arbre, augmentant ainsi nos chances de nous situer dans la partie la plus complète, celle où l'on risque le moins de devoir créer un noeud fantôme.

Au-delà de $k_{min} - 2$, on ne calcule donc plus les extensions maximales. Par défaut, un noeud sans extension maximale se verra attribuer un $MaxExt = \infty$. Ainsi, lors d'une descente de récursion, si un des fils du noeud courant produit un $MaxExt = \infty$, l'extension maximale de ce noeud courant prend aussi cette valeur, et tous les parents du noeud recueilleront cette valeur. Dans ce cas, ces noeuds ne pourront apporter aucune optimisation ultérieure.

Notons que la racine de l'arbre $\mathcal{A}$ recueillera probablement, au terme de l'extraction, une extension maximale valant l'infini (à moins qu'aucun modèle n'ait atteint les longueurs requises). Si ces propagations de valeurs inutilisables entraînent des calculs superflus, nous tablons sur le fait que ces calculs sont peu coûteux, puisque réalisés dans le cours normal de l'algorithme qui effectue naturellement un parcours préfixe.

5.3.3.3 Coût de création d'un noeud fantôme

La question du coût de la création d'un noeud fantôme est importante pour nous, car elle détermine si l'utilisation systématique (quand c'est possible) des $MaxExt$ est judicieuse ou non. Nous savons maintenant que plus on descend dans l'arbre, moins il est complet et plus on risque de devoir créer un noeud fantôme pour stocker le $MaxExt$ d'un modèle. En même temps, plus on se rapproche de k_{min}, moins le gain éventuel procuré par le lemme 13 est important.

Nous n'avons pas à ce jour évalué le taux de noeuds fantômes créés lors d'un parcours d'extraction donné. On peut cependant calculer le coût de création d'un noeud fantôme. Dans le pire des cas, il faut créer deux noeuds : celui qui casse un arc pour venir remplacer un pseudo-noeud, et un second noeud, la branche les reliant correspondant au suffixe manquant du modèle concerné. Ces deux créations peuvent être facilement réalisées en temps constant. On peut en effet, pendant le parcours de l'arbre des modèles, pointer en permanence sur l'équivalent du modèle en cours dans l'arbre des suffixes, ou sur le dernier noeud visité, dans le cas où cet équivalent n'existe pas. Il suffit alors de créer les deux noeuds nécessaires, en temps constant.

Le coût en temps d'un tel stockage n'est donc pas significatif. Le coût en espace doit par contre être mesuré attentivement à l'aide d'expérimentations, puisqu'en créant des noeuds fantômes on brise la linéarité de l'arbre des suffixes.

Dans le cas où une telle méthode de stockage s'avérerait trop coûteuse à l'usage, il faudrait envisager d'autres techniques.

5.3.4 Utiliser les extensions maximales

Considérons maintenant que tous les $MaxExt$ des noeuds correspondant à des modèles ayant déjà été visités ont été calculés, et ce jusqu'au niveau $k_{min} - 2$ inclus.

Nous traitons le modèle m de longueur $k < k_{min}$, que nous cherchons à étendre. Il faut d'abord remarquer que seuls les suffixes de m lui étant lexicographiquement inférieurs peuvent lui fournir un $MaxExt$, les autres n'ayant pas encore été visités. Le modèle m n'ayant évidemment pas encore de $MaxExt$ à cet instant, on commence donc par suivre son lien-suffixe vers son plus grand suffixe s. Si celui-ci est lexicographiquement supérieur à m on poursuit la descente des liens-suffixes à la recherche d'un autre suffixe lexicographiquement inférieur. Si la racine est atteinte, on n'a rien pu tirer des $MaxExt$ des suffixes de m.

Si on atteint un suffixe lexicographiquement inférieur à m, il possède nécessairement un $MaxExt$ puisqu'il a déjà été visité. On applique alors la formule du lemme 13 ; si elle est vérifiée, on peut alors interrompre l'extension de m. Sinon, il est inutile de rechercher un autre suffixe ayant un $MaxExt$ plus petit, grâce au lemme suivant :

Lemme 14
Soit un modèle $m \in \Sigma^+$. Pour tout suffixe s de m, on a $MaxExt(m) \leq MaxExt(s)$.

Preuve : Soit $MaxExt(m) = k$. Prenons pour hypothèse qu'il existe un suffixe s de m tel que $MaxExt(s) < k$. Cela signifie qu'il existe un $v \in \Sigma^j, j < k$ et pas de mot plus long, tel que le modèle sv soit valide. Or nous savons qu'il existe un $u \in \Sigma^k$ tel que mu a des occurrences dans au moins q séquences. Le mot su, étant un suffixe de mu, a également des occurrences dans au moins q séquences. Le mot u a une longueur k et l'hypothèse est contredite.□

La séquence d'opérations à effectuer avant toute extension du modèle m pour s'enquérir d'un $MaxExt$ qui serait profitable est la suivante ($LS(x)$ représente le noeud pointé par le lien-suffixe [3] de x, donc le noeud qui correspond au plus grand suffixe de x) :

$s = m$
Faire
  $s = LS(s)$
Tant que s `est lexicographiquement supérieur à` m
Si $|m| + MaxExt(s) < k_{min}$ **Alors**
  `Interrompre l'extension de` m
FinSi

Dans le pire des cas, on parcourt tous les suffixes de m jusqu'à la racine. Le coût ajouté à chaque visite de modèle est donc proportionnel à sa longueur. On peut imaginer un prétraitement de l'arbre des suffixes de façon à ce que chaque lien-suffixe pointe non plus sur son suffixe le plus grand mais sur son suffixe le plus grand lui étant lexicographiquement inférieur. Ceci fait, le coût de la recherche d'une optimisation par extension maximale serait constant.

Autre optimisation envisageable : étant donné que les valeurs infinies des extensions maximales se propagent nécessairement jusqu'à la racine, on pourrait simplement mémoriser la longueur du modèle le plus petit n'ayant pas reçu une extension maximale infinie. Cela permettrait, lors de la recherche d'une optimisation pour un modèle, d'interrompre cette optimisation à moindre coût si ce seuil est atteint, signifiant qu'on ne trouvera plus de suffixes de ce modèle ayant des extensions maximales utilisables.

En respectant cet algorithme, un problème se pose pourtant : les pseudo-noeuds n'ont pas de lien-suffixe. Dans l'arbre compact, seuls les noeuds possèdent de tels liens. Un modèle pour lequel on cherche une optimisation peut correspondre à un pseudo-noeud. Dans ce cas, ce pseudo-noeud ne sera transformé en noeud-fantôme qu'ultérieurement, après le parcours de son sous-arbre. Or, auparavant, pour essayer d'optimiser ce modèle, il nous faut atteindre les noeuds correspondant à ses suffixes. Pour atteindre le plus grand d'entre eux, nous devrons donc nous résoudre à suivre les liens-suffixes du noeud parent le plus proche. Et à répéter l'opération tant que le noeud atteint est un pseudo-noeud. N'oublions cependant pas que, en suivant les contraintes que nous nous imposons, la rencontre d'un pseudo-noeud doit être relativement rare.

Il est important de se souvenir que l'extension d'un modèle m d'un niveau nécessite Σ extensions de toutes les occurrences de m et autant de calculs de couverture des ensembles d'occurrences obtenus. Il peut donc être intéressant de profiter des extensions maximales pour éviter ne serait-ce que la dernière extension d'un modèle. Jusqu'à maintenant, les extensions maximales nous servent à prévoir qu'aucune extension d'un modèle n'atteint la

[3] comme dans l'algorithme 2 d'extraction de modèles structurés, l'utilisation des extensions maximales nécessite que les liens-suffixes placés dans l'arbre lors de sa construction soient conservés une fois celle-ci terminée.

longueur minimum. Lorsque le $MaxExt$ recueilli fait état d'une extension maximale nous amenant à une longueur supérieure ou égale à k_{min}, nous abandonnons toute possibilité de court-circuiter l'extension.

Or nous pouvons tirer une autre information utile de $MaxExt$: si, pour un modèle m, on recueille un $MaxExt$ indiquant que la plus grande extension de m amène à une longueur comprise entre k_{min} et $k_{max} - 1$, alors on peut conserver cette valeur dans les extensions futures de m (elle sera éventuellement revue à la baisse lors des extensions ultérieures de m), ceci afin d'éviter certaines des dernières extensions de m.

Exemple : $k_{min} = 7, k_{max} = 10$. Lors de la visite du modèle $AGATA$, on trouve que $MaxExt(ATA) = 3$. Cela signifie que la plus grande extension de $AGATA$ donnera un modèle de longueur 8. On ne peut pas appliquer le lemme 13, mais on peut conserver cette valeur pour les extensions futures du modèle. On étendra donc $AGATA$ à $AGATAA$, $AGATAC, \ldots$ en se souvenant qu'à la longueur 8 il est inutile de poursuivre. Si certains chemins de l'arbre des modèles s'arrêteront évidemment avant ce niveau ($MaxExt$ indique l'extension maximale), d'autres pourront bénéficier de cette information et éviter ainsi une dernière extension inutile et coûteuse.

Cette amélioration n'apporte aucun surcoût en espace et en temps, puisqu'il suffit de propager une valeur lors de la descente récursive.

Pour exploiter au mieux cette idée, il faudrait cependant que nous calculions les extensions maximales des modèles de longueur comprise entre $k_{min} - 1$ et $k_{max} - 1$, ce qui, au stade où en sont nos recherches, risque de s'avérer coûteux du fait de l'augmentation du nombre de noeuds fantômes à créer au fur et à mesure qu'on descend dans l'arbre. Nous nous contentons donc pour le moment de profiter, quand cela est possible, de cette opportunité en conservant la structure « économique » de stockage des extensions maximales. Si nos travaux nous amènent à découvrir un moyen raisonnable de stocker toutes les extensions maximales sans limitation de profondeur, nous pourrons alors profiter pleinement de cette seconde optimisation.

5.3.5 Extension à plusieurs boîtes

Dans le cas d'un modèle structuré composé de deux boîtes, la seconde boîte peut profiter des extensions maximales des premières boîtes déjà calculées. Ceci à la condition que le contexte dans lequel l'extraction de la première boîte a été faite soit moins contraint que celui de la seconde : il faut par exemple que la longueur et le nombre de substitutions de la première boîte soient supérieurs ou égaux à ceux de la seconde. On peut de cette façon rechercher pour une seconde boîte un $MaxExt$ intéressant sur le même principe que précédemment, en veillant à ce que le suffixe de cette seconde boîte en cours d'extension et dont on attend un $MaxExt$ soit lexicographiquement inférieur à la première boîte.

Les premières boîtes ne peuvent en aucun cas bénéficier d'extensions maximales qui seraient calculées lors de la visite des secondes boîtes ; celles-ci n'auraient aucun sens puisqu'elles sont spécifiques au contexte « seconde boîte ».

Un des problèmes qui se posent concerne les liens-suffixes : comment atteindre le modèle correspondant au plus grand suffixe d'une seconde boîte.

Si on adopte un arbre dynamique des suffixes comme dans l'algorithme 2, cela ne pose pas de problème puisque toutes les extractions se déroulent dans les niveaux hauts de l'arbre, comme si chaque boîte était indépendante des précédentes. Il suffira donc de suivre le lien-suffixe de la seconde boîte pour obtenir le suffixe souhaité.

Dans le cas de l'algorithme 1, la démarche pose plus de problèmes étant donné que les extensions des boîtes qui suivent la première se déroulent dans le bas de l'arbre, là où aucune extension maximale n'est stockée. Une solution simple consiste à suivre l'extraction de la seconde boîte depuis le haut de l'arbre, en pointant sur le noeud-occurrence de cette seconde boîte seule, et à dérouler les liens-suffixes depuis ce pointeur quand c'est nécessaire. L'extension à plus de deux boîtes n'apporte pas de difficultés supplémentaires.

Une autre approche plus pointue consisterait à recommencer à zéro le travail de calcul des extensions maximales dans le contexte « seconde boîte », pour chaque première boîte trouvée.

5.3.6 Quelques résultats et conclusions

Dans le but d'évaluer l'importance de l'optimisation que nous décrivons dans cette section, nous avons réalisé quelques tests simples dans des conditions variées, sur des modèles composés d'une boîte.

Ces tests consistent simplement à conserver la liste des modèles visités lors d'une extraction avec SMILE, puis à post-traiter cette liste en calculant les extensions maximales des modèles, et enfin à rechercher les modèles dont on aurait pu éviter la visite grâce à l'application du lemme 13. La seconde optimisation qui consiste à éviter la dernière extension de certains modèles n'a pas été prise en compte.

Un autre point n'a pas été considéré, qui fait que ces résultats sont probablement légèrement meilleurs que ceux que nous obtiendrons en réalité. En effet, un modèle dont l'extension est interrompue par l'utilisation de l'extension maximale d'un de ses suffixes, reçoit une extension maximale déduite de celle apportée par ce suffixe. Le contexte du modèle en question étant plus restreint que celui de son suffixe, il aurait pourtant pu, si l'on avait effectué un parcours normal dans son sous-arbre, obtenir une extension maximale inférieure. Il y a donc une perte de précision des extensions maximales lors de leur utilisation, qui se propage à chaque fois qu'une optimisation est réalisée.

Le tableau 5.1 présente les résultats obtenus. Ces quelques tests permettent d'observer un résultat prévisible : un faible nombre de modèles valides atteignant la longueur requise indique que la majorité des modèles de la longueur requise ont été invalidés (et donc qu'on a pu calculer leurs extensions maximales), les optimisations sont donc nombreuses dans ce cas. L'utilisation des extensions maximales permet d'obtenir plus rapidement des disqualifications de modèles. D'où les bons résultats lors des extractions qui produisent peu de modèles.

q	e	k	obtenus	visités	évitables	% optimisation
20	1	8	4472	55 553	795	1.5
20	0	6	103	2 853	56	2
20	1	9	534	73 441	14 644	20
20	0	7	4	3 265	896	27
10	1	10	569	180 473	54 727	30
10	0	8	4	7 125	2 683	38
5	1	11	484	419 255	168 339	40
20	1	10	0	75 577	36 203	48
20	0	8	0	3 281	1 753	54

Tab. 5.1 – Résultats de quelques tests sur les extensions maximales. Le jeu de données utilisé est issu du génome de *Bacillus subtilis* et contient 1076 séquences pour un total de 400Kb. q indique le quorum choisi, e le nombre de substitutions, et k la longueur des modèles extraits. La colonne *obtenus* reporte le nombre de modèles valides et de longueur k produits par SMILE.

Nous constatons que le nombre de modèles évitables atteint parfois les 50%. Cette optimisation peut donc s'avérer très fructueuse dans certains cas. Nous en concluons que la notion d'extension maximale mérite d'être explorée plus avant et éventuellement implantée.

Ce constat positif ouvre la porte à de nombreux travaux ; il reste à déterminer une borne raisonnable au coût de l'exploitation des extensions maximales, et à les implanter dans SMILE afin de mesurer les bénéfices réels de cette optimisation.

Si les extensions maximales posent encore nombre de problèmes d'exploitation dans l'absolu, nous pensons que, dans des conditions d'utilisation restreintes telles que nous les avons décrites, elle doivent s'avérer assez économiques et intéressantes lors de certaines extractions.

Notons pour finir que les algorithmes que nous manipulons ne se prêtent pas parfaitement à cette optimisation, étant donné qu'ils sont fondés sur un parcours en profondeur de l'arbre virtuel des modèles. Un parcours en largeur permettrait de profiter de façon optimale des extensions maximales, puisque tous les suffixes d'un modèle donné auraient déjà été visités quand on atteindrait ce modèle. Mais ce parcours en largeur, de par l'empilement important d'occurrences qu'il nécessite, n'est pas envisageable dans l'algorithme actuel, qui devrait donc être reconsidéré.

Nos travaux, visant à obtenir une structure de données moins coûteuse pour les algorithmes que nous utilisons, doivent aussi s'orienter vers la recherche d'une structure de données facilitant l'utilisation des extensions maximales.

5.4 Discussion

Les objectifs théoriques que nous nous étions fixés ont été respectés. Nous avons en effet allégé la structure de l'arbre, défini de nouveaux modèles possédant des critères qui devraient les rendre plus à même de décrire les sites de fixation qui nous intéressent, et développé deux algorithmes d'extraction pour ces modèles structurés, basés sur des critères structurels plus forts que ceux utilisés habituellement. Ces algorithmes conservent les avantages de ceux dont ils s'inspirent, à savoir la linéarité en la taille du jeu de séquences à traiter.

Les modèles structurés nous permettent maintenant de décrire un ensemble de sites de fixation associés par des contraintes plus ou moins souples. Nous verrons dans le chapitre 7 que cette description plus proche de certains modèles biologiques observés nous permet de détecter certains sites indétectables si l'on considère des modèles simples et indépendants.

Les algorithmes décrits dans ce chapitre se basent sur l'algorithme original de Marie-France Sagot. Ils supportent donc la même contrainte intrinsèque que ce dernier : le nombre de substitutions reste un facteur exponentiel dans les complexités. Ces algorithmes se destinent donc en théorie à l'extraction d'un certain type de modèles structurés, pas trop longs et relativement bien conservés. Les complexités dans le pire des cas exprimées sont cependant des bornes très larges, et on observe en pratique que ces algorithmes résistent assez bien à l'extraction de modèles mal conservés. Nous montrons dans les chapitres qui suivent que nos algorithmes exacts s'avèrent efficaces et soutiennent souvent la comparaison en terme de temps de calcul avec les algorithmes d'optimisation heuristiques les plus utilisés. Les optimisations réalisées à l'aide des extensions maximales nous laissent de plus espérer des améliorations importantes dans certains cas. En outre, au terme de ces travaux, il nous apparaît que nombre d'optimisations sont probablement encore possibles pour visiter mieux l'espace des solutions de notre problème, ce dont nous parlerons au terme de ce mémoire.

Nous disposons maintenant d'algorithmes nous permettant de réaliser l'extraction de modèles sur des critères structurels qui s'approchent de ceux des sites de fixation que nous souhaitons détecter. Pour réaliser correctement l'inférence de ces sites, il reste cependant à appliquer des critères de signification aux modèles extraits. Notre choix est de réaliser cette évaluation dans une seconde phase, en post-traitement des résultats obtenus lors des extractions. Cette seconde phase d'évaluation ne fait pas véritablement partie de nos recherches au même titre que les algorithmes que nous avons développés : les méthodes employées étant aux limites de notre domaine de compétences, nous nous sommes contentés d'utiliser et d'implanter des techniques statistiques connues. Nous avons également collaboré avec des mathématiciens en vue d'obtenir des formules de probabilité d'apparition adaptées à notre problème, c'est-à-dire pour des modèles structurés avec erreurs.

Plusieurs méthodes d'évaluation ont finalement été intégrées dans l'outil que nous avons développé. Nous les décrivons dans la section 6.1.3 page 150 du chapitre suivant.

En conclusion, il nous faut insister sur le fait que les algorithmes que nous avons développés sont les seuls algorithmes à ce jour, exacts ou non, qui permettent l'extraction de consensus aussi souples, c'est-à-dire composés d'un nombre quelconque de boîtes séparées par des *spacers* de longueur comprise dans un intervalle quelconque, sans limitations théoriques du nombre de substitutions ni de la longueur des boîtes.

Le chapitre suivant est consacré à la description de l'outil que nous avons développé autour de ces algorithmes, et à quelques expériences visant à donner un aperçu de ses capacités. Le chapitre 7 rassemble des résultats obtenus lors d'expériences effectuées sur des données biologiques, et qui permettent de mieux cerner les capacités et limites des modèles et algorithmes que nous avons développés. Comparativement aux outils existants, nous verrons que nos algorithmes s'avèrent capables de traiter de plus grandes quantités de données, et d'extraire des signaux plus faiblement représentés grâce au gain de spécificité apporté par les modèles structurés.

Chapitre 6

SMILE

Suite logique aux développements algorithmiques du chapitre précédent, nous nous sommes attachés à implanter les structures, représentations et méthodes que nous avons élaborées. Avant d'être utilisée à des fins biologiques, cette implantation a d'abord permis de valider l'efficacité des algorithmes, dont les complexités, appliquant des majorations très larges, ne permettent pas de prédire le comportement en pratique. Ce chapitre décrit le programme que nous avons développé, dénommé SMILE [1], qui est un outil d'inférence permettant d'extraire des modèles simples ou structurés sur des critères structurels, et d'estimer ensuite la signification des modèles obtenus au moyen de différentes mesures de signification.

De même que les algorithmes qu'il utilise, SMILE est le seul outil disponible à ce jour qui permette l'inférence de modèles structurés composés d'un nombre quelconque de boîtes, sur un alphabet quelconque et sans limitations sur les caractéristiques structurelles des modèles. Il est à ce titre entièrement générique, comme nous le souhaitions en débutant nos travaux, et peut donc être appliqué à l'extraction ou à l'inférence de modèles structurés dans n'importe quel type de séquences, y compris protéiques.

Dans la première section, nous décrivons l'architecture de SMILE et détaillons tour à tour les différents modules qui le composent. Entre autres, nous présentons le module d'évaluation et les méthodes que nous avons utilisées afin d'estimer la signification des modèles extraits par nos algorithmes.

Puis, dans la section 6.2, nous présentons des exemples d'extractions accompagnées de mesures de performances.

Dans la section 6.4, nous discutons du positionnement de SMILE par rapport aux outils existants, au vu des pré-requis qu'il impose et de ses performances.

Enfin, dans la section 6.3, nous passons en revue les modifications et améliorations qui restent à apporter à SMILE à ce jour.

Un article concernant SMILE est en cours de rédaction à ce jour ([Marsan et al., 2002]).

[1] au-delà de sa connotation sympathique, le choix du nom de SMILE a été motivé par les initiales de Structured Motifs Inference and Evaluation, la signification du L étant laissée à l'appréciation de l'utilisateur !

6.1 Description

SMILE est un outil qui extrait d'un jeu de séquences quelconques des modèles respectant un certain nombre de critères structurels fournis, et qui évalue ensuite la signification des modèles obtenus. SMILE est donc composé de deux sous-parties :
- la première agit comme un filtre exact, elle réalise une extraction dans un jeu de séquences donné, à partir des critères structurels fournis par l'utilisateur. C'est dans cette partie que sont implantés les algorithmes du chapitre précédent. Au terme de cette étape, on dispose de tous les modèles respectant les critères choisis ;
- la seconde traite les modèles produits par la première et leur associe des valeurs de signification, calculées selon différentes mesures, statistiques ou probabilistes.

La figure 6.1 illustre l'architecture employée. Nous traitons tour à tour dans cette section de chacun des éléments de cette figure : nous décrivons d'abord l'interface et les critères à fournir au programme, puis le sous-programme d'extraction, et enfin le sous-programme d'évaluation et les différentes mesures qu'il utilise pour classifier les modèles.

SMILE a été développé en C (sauf l'interface, en Perl) sous Unix. Ses sources sont disponibles sur demande (`marsan@univ-mlv.fr` ou `sagot@biomserv.univ-lyon1.fr`). Cette section a pour but de présenter SMILE, et ne se veut donc en aucun cas être une documentation pour l'utilisateur. Des explications et des exemples commentés sont disponibles dans le *package* que nous avons développé.

6.1.1 Interface

Pour exécuter SMILE, il suffit de fournir un fichier de paramètres contenant les critères structurels d'extraction choisis, ainsi que la méthode d'évaluation qu'on souhaite éventuellement appliquer aux modèles extraits. Nous détaillons ici les critères structurels que SMILE peut prendre en compte.

6.1.1.1 Format des séquences et alphabets

Dans le but de traiter des données biologiques, SMILE utilise des séquences au format FASTA, pouvant être écrites sur un alphabet quelconque. On peut donc réaliser des extractions dans des séquences nucléiques, protéiques ou autres. Comme expliqué dans la section 5.1.1 page 100, la structure d'arbre des facteurs que nous utilisons est linéaire en espace et affiche en pratique une constante de l'ordre de 100. L'unique limitation pour la taille du jeu de données à fournir provient donc de la mémoire disponible.

L'alphabet sur lequel SMILE construit les modèles extraits est également modifiable. On peut choisir d'exprimer des relations d'équivalence entre certains symboles ou d'en ignorer, ou bien encore d'utiliser des jokers. Par exemple, sur un jeu de séquences d'ADN contenant les symboles de base A, C, G, T et le symbole N, on peut décider que l'extraction

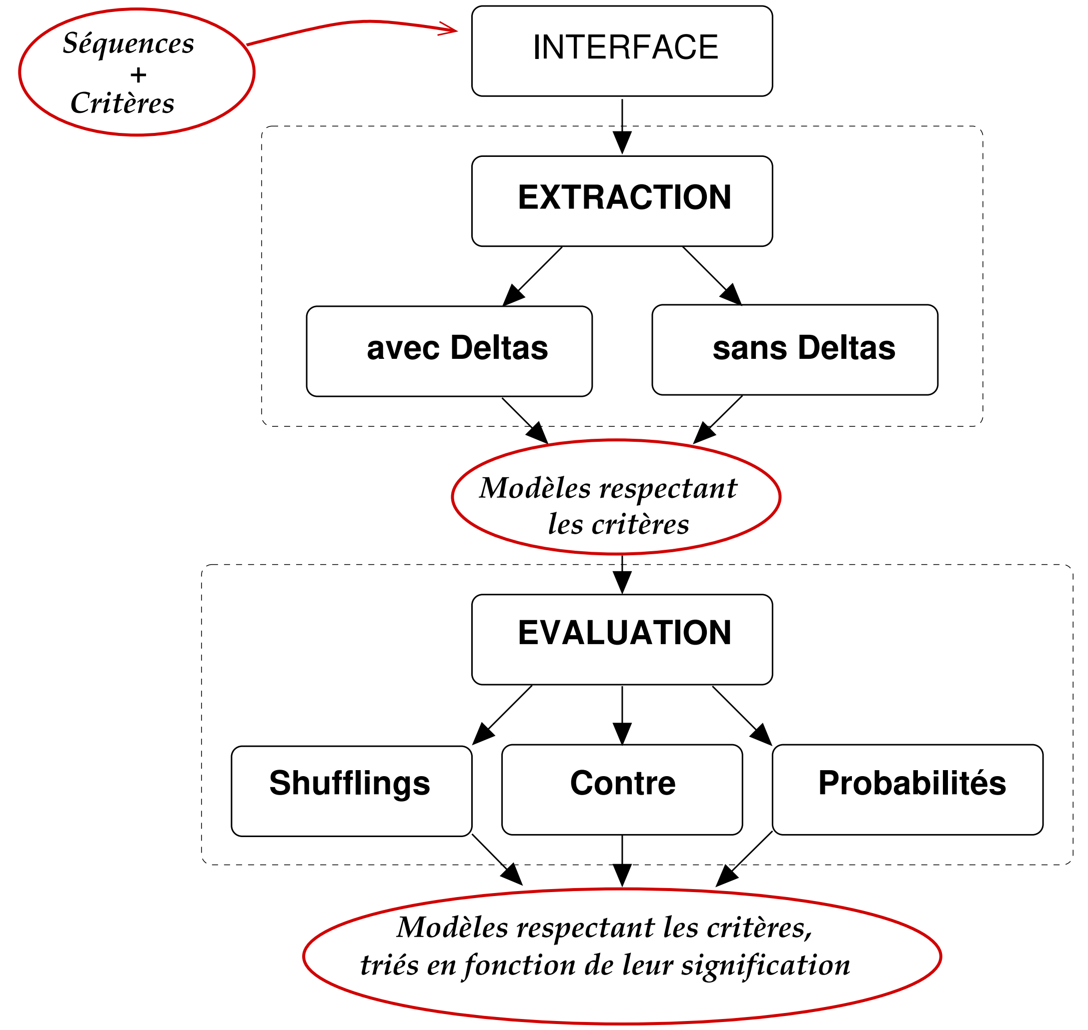

FIG. 6.1 – L'architecture de SMILE.

ne produise que des modèles sur l'alphabet R, Y et n'utilise pas d'occurrence contenant un N pour construire ces modèles.

6.1.1.2 Critères obligatoires des modèles

Certains critères comme le quorum, les longueurs des boîtes, le nombre de substitutions ou les sauts entre boîtes, sont nécessaires à l'extraction des modèles. En pratique, il faut donc avoir *a priori* une certaine connaissance des modèles que l'on souhaite extraire. Dans le cas contraire, les bonnes performances de SMILE et l'utilisation de fourchettes pour certains critères (longueurs, sauts, deltas) permettent tout de même de « tâtonner » à la recherche d'un modèle dont on ne connaît pas la structure.

Certains critères, globaux, portent sur l'ensemble d'un modèle structuré, tandis que d'autres, locaux, ne concernent qu'une des boîtes du modèle. Critères globaux et locaux sont donc en concurrence. Après avoir vérifié qu'ils ne sont pas en contradiction, SMILE utilise simultanément ces deux types de critères lors de l'extraction, ce qui permet, dans une certaine mesure, d'exprimer des corrélations entre plusieurs boîtes.

Dans le cas d'une extraction de modèles simples, la distinction entre critères locaux et globaux n'a pas de sens.

Quorum

Le quorum indique le nombre minimum de séquences qu'un modèle doit recouvrir pour être valide. Le choix du quorum nécessite une certaine connaissance du jeu de données. Un jeu très propre, dont on sait par exemple que le modèle à extraire est présent dans chaque séquence, pourra être utilisé avec un quorum élevé, alors qu'un jeu brut où le modèle est peu représenté demandera un quorum faible.

Comme nous l'avons vu dans les deux chapitres précédents, c'est grâce au quorum qu'on peut éviter d'étendre certains modèles dont on sait à l'avance qu'ils ne produiront pas de résultats (lemme 2 page 86). Ce paramètre joue donc un rôle crucial dans le fonctionnement de l'algorithme, précisément sur la profondeur à laquelle les modèles inintéressants vont être étendus avant d'être éliminés.

Faute d'information sur la couverture du modèle qu'on souhaite extraire, on peut évidemment choisir par défaut un quorum très bas au risque de recueillir beaucoup de modèles. Rappelons qu'un des buts de l'extraction est de produire un jeu de modèles de taille raisonnable à fournir aux outils d'évaluation qui viennent en aval. Certaines méthodes réalisent une évaluation directe de tous les modèles possibles sans filtrage préalable sur des critères structurels (seule leur longueur est fixée, voir les outils développés par Van Helden [van Helden et al., 1998]) mais ce type d'approche, générant toutes les solutions possibles, oblige à se restreindre à des modèles de petite taille, autorisant peu ou pas de substitutions, et exprimés sur un alphabet simple. Notre approche consiste au contraire à considérer des modèles plus complexes et à filtrer ceux qui répondent à certains critères structuraux forts, dont le quorum, avant d'évaluer leur signification. Diminuer exagérément le quorum revient à assouplir les contraintes sur les modèles et donc, dans le pire des cas, à parcourir l'ensemble de l'espace des solutions, ce que nous souhaitons éviter.

C'est la raison pour laquelle le quorum, clé de voûte de l'algorithme, doit être choisi consciencieusement, quitte à relancer l'exécution avec différentes valeurs. Il existe en général, en fonction du jeu de données qu'on utilise, deux seuils de quorum : le premier au-dessus duquel on n'obtient aucun modèle, et le second en dessous duquel on obtient tous les modèles possibles. Hors de cet intervalle (*a priori* inconnu), l'algorithme d'extraction n'est d'aucune utilité. Le dépassement du premier seuil provoque cependant des extractions très rapides (tous les modèles étant éliminés en haut de l'arbre) alors qu'un quorum plus faible que le second seuil demande un parcours de tout l'arbre virtuel des modèles jusqu'aux longueurs choisies. Si actuellement ce tâtonnement échoit à l'utilisateur, la déduction, à partir des séquences fournies, des seuils de quorum pertinents (avec éventuellement à la clé l'élimination de ce paramètre) fait partie des questions à étudier.

Longueur
Parmi les critères dont SMILE a besoin pour réaliser une extraction figurent les longueurs de chacune des boîtes composant les modèles. Dans le cas où ce critère n'est pas connu précisément, on peut fournir une fourchette dont la borne supérieure peut être infinie. Dans ce cas, l'extension d'un modèle ne s'interrompt que quand il ne respecte plus le quorum.

Il faut également préciser une fourchette de longueur globale pour l'ensemble des boîtes (dont la borne supérieure peut également être infinie). Ce critère permet d'introduire une certaine corrélation entre les boîtes extraites : l'extraction d'un modèle de deux boîtes de longueurs 6 à 8 avec une longueur globale de 14 produira des modèles composés de boîtes de longueur 6 et 8, 7 et 7, ou 8 et 6.

Substitutions
Chaque boîte doit posséder un nombre de substitutions maximal propre. On doit également exprimer un nombre maximum de substitutions pour l'ensemble des boîtes qui composent le modèle. Cela permet aussi une certaine corrélation entre boîtes : l'extraction de modèles de deux boîtes acceptant chacune une substitution, avec un maximum global d'une substitution, produira des modèles dans lesquels une substitution sera jouée dans l'une ou l'autre des deux boîtes.

Sauts
Dans le cas d'une extraction de modèles structurés, l'utilisateur doit spécifier des longueurs pour les *spacers* qui séparent les boîtes successives. Si ces longueurs sont mal connues, SMILE permet l'utilisation de fourchettes.

6.1.1.3 Critères optionnels des modèles

Certains critères structurels supplémentaires, optionnels, permettent si besoin est d'affiner la description des modèles à extraire.

Deltas
L'utilisation d'une fourchette de saut entre deux boîtes permet d'extraire des modèles structurés autorisant une certaine souplesse pour ce critère. En élargissant cette fourchette, on recueille nécessairement de plus en plus de modèles, sans pour autant savoir si une partie de cet intervalle est plus pertinente qu'une autre.

Comme expliqué dans le chapitre précédent, les deltas donnent à l'utilisateur un moyen d'exprimer la taille des intervalles de saut à considérer à l'intérieur de la fourchette spécifiée. Pour un saut entre deux boîtes données, ils génèrent plusieurs intervalles de validité pour lesquels les quorums seront calculés séparément. Ainsi, les modèles produits seront séparés en fonction des intervalles sur lesquels ils sont valides.

Composition
SMILE permet de fixer la composition maximale en un ou plusieurs symboles, pour une boîte précise et/ou pour l'ensemble du modèle.

6.1.1.4 Exemples de fichiers de paramètres

Pour réaliser une extraction avec SMILE, la première chose à faire consiste à définir l'alphabet sur lequel les modèles seront produits. On écrit donc un fichier d'alphabet qui sera nommé comme réfèrence dans le fichier de paramètres de l'extraction.

Un fichier type pour exprimer des modèles sur l'alphabet des nucléotides de base contiendrait le texte suivant :

```
Type:Nucleotides
A
C
G
T
```

Si l'on souhaite effectuer une extraction sur un alphabet plus large, en ajoutant les symboles R et Y des purines et pyrymidines, ainsi que le joker N, à l'alphabet des modèles générés, on écrira le fichier alphabet suivant :

```
Type:Nucleotides
A
C
G
T
AG
CT
*
```

Le symbole $*$ indique que l'on utilise des jokers pour générer les modèles. L'introduction de symboles très dégénérés va de pair avec une limitation de leur utilisation dans les modèles (par le biais du critère de composition) si on veut éviter de générer des modèles trop laxistes. En autorisant le joker sans limitation, on produira à coup sûr des modèles de la forme $NNNNN\dots$

Le type précisé dans les fichiers d'alphabet (`Nucleotides`, `Proteins` ou `Unknown`) permet à SMILE de reconnaître l'alphabet utilisé et de remplacer automatiquement un ensemble de symboles reconnu par le symbole IUPAC correspondant. Sur l'alphabet précédent, le modèle `CA[AG][ACGT]G` sera ainsi traduit en `CARNG`.

Une fois le fichier d'alphabet des modèles défini, on peut écrire un fichier de paramètres pour l'extraction. Dans le cas de modèles simples, le fichier suivant réalise par exemple l'extraction de modèles de longueur comprise dans l'intervalle $[14; +\infty[$, avec au plus trois substitutions, apparaissant dans au moins 90% des séquences du fichier `seq.fasta` :

```
EXTRACTION (Step 1) ==============================
FASTA file                        seq.fasta
Output file                       test.smile

GLOBAL PARAMETERS =============
Alphabet file                     dnabase.alphabet
Quorum                            90
Total min length                  14
Total max length                  0
Total substitutions               3
Boxes                             1
```

`test.smile` est le fichier où seront écrits les modèles extraits, et `dnabase.alphabet` est le fichier alphabet, du type de ceux décrits plus haut.

L'extraction de modèles structurés composés de deux boîtes de longueurs respectivement comprises dans les intervalles $[4; 5]$ et $[6; 8]$, avec deux substitutions au total dont une au plus dans la première boîte, un intervalle de saut $[10; 20]$ et un $\delta = 1$, apparaissant dans au moins 10% des séquences du fichier `fasta`, et ayant au plus un joker dans une des deux boîtes, est réalisée en écrivant le fichier de paramètres suivant :

```
EXTRACTION (Step 1) ==============================
FASTA file                      seq.fasta
Output file                     test.smile

GLOBAL PARAMETERS =============
Alphabet file                   dnabase.alphabet
Quorum                          10
Total min length                10
Total max length                13
Total substitutions             2
Boxes                           2
Composition in *                1

BOX 1 ================
Min length                      4
Max length                      5
Substitutions                   1
Composition in *                1
Min spacer length               10
Max spacer length               20
Delta                           1

BOX 2 ================
Min length                      6
Max length                      8
Substitutions                   2
Composition in *                1
```

Avec ces deux exemples, nous présentons des requêtes très différentes ; la première, très souple, n'impose qu'un quorum, une longueur minimale, et un nombre de substitutions maximal. La seconde, très pointue, définit un modèle structuré précis dont on connaît visiblement certaines propriétés.

6.1.2 Extraction

Nous décrivons ici l'implantation des algorithmes d'extraction, l'influence de certains critères sur une extraction, et les sorties produites.

6.1.2.1 Description

Le sous-programme d'extraction traite la requête issue d'un fichier de paramètres du type de ceux décrits précédemment, et produit un fichier de sortie contenant les modèles qui respectent les critères spécifiés. Il permet indifféremment l'extraction de modèles simples ou structurés. Deux sous-programmes distincts sont exécutés selon que l'on utilise les deltas ou non, ceci à des fins d'optimisation. La structure de données utilisée est bien sûr l'arbre des facteurs du chapitre précédent, dont la profondeur maximale est calculée en fonction des critères des modèles à extraire. L'extraction de modèles sans limite de longueur maximale entraînera la construction de l'arbre des suffixes classique.

Le programme utilise l'algorithme 1 du chapitre précédent, qui réalise des sauts dans l'arbre pour traiter les *spacers*. Cet algorithme a été choisi dans un premier temps car moins complexe à implanter à partir de l'algorithme original de Marie-France Sagot (que nous avions implanté aux débuts de nos travaux). L'algorithme 2 nécessite une plus grande refonte des structures de données. Les optimisations apportées par les extensions maximales, étudiées récemment, n'ont pas encore été incorporées.

Parmi les comportements propres à l'algorithme que nous utilisons, il est à noter que, puisque les modèles sont étendus par la droite, les performances de l'extraction d'un modèle structuré composé de deux boîtes dépend de l'ordre de ces boîtes : si l'on donne des critères faibles à la première boîte et des critères forts à la seconde, l'extraction sera plus coûteuse que dans le cas contraire. En effet, plus les contraintes fortes sont rapidement appliquées, plus on élimine un grand nombre de modèles prématurément. Bien entendu, ce comportement inhérent à la conception de nos algorithmes ne peut en aucun cas conditionner les critères d'extraction de l'utilisateur, mais il peut être utile d'en avoir connaissance pour exploiter au mieux le programme. Ce comportement devrait être atténué par l'utilisation de l'algorithme 2, puisque l'extraction de toutes les boîtes débute avec une seule occurrence.

SMILE est un outil d'extraction générique mais n'a cependant pas la prétention d'extraire tous les types de signaux : certains d'entre eux, très longs et dégénérés (comme les signaux ALU), nécessitent d'autres approches algorithmiques. L'extraction de signaux très longs avec un nombre modéré de substitutions ne pose cependant pas de problèmes.

Les limites d'utilisation dont nous parlons dans cette section ne sont pas algorithmiques : SMILE n'impose pas de contraintes sur les critères fournis. Mais de ces critères dépend la taille de l'espace des solutions : des critères trop permissifs provoqueront un temps d'extraction prohibitif, et produiront un grand nombre de modèles sans intérêt. Le temps de traitement de ces modèles par le sous-programme d'évaluation éventuellement utilisé se verra aussi augmenté en conséquence.

Les répercussions de la modification de certains paramètres observées lors de nos expérimentations sur le fonctionnement de l'extraction nous donnent des indications sur les compromis à faire entre sensibilité et efficacité :

- **l'alphabet des modèles** influe sur la largeur de l'arbre virtuel des modèles. Le parcours en profondeur effectué dans cette structure est rendu plus long par l'augmentation du cardinal de l'alphabet, car il existe potentiellement plus de modèles à visiter. L'utilisation d'un alphabet trop dégénéré sans restriction de composition sur les symboles dégénérés produit un grand nombre de modèles valides ayant beaucoup d'occurrences, et peut de ce fait ralentir considérablement l'algorithme ;

- **le quorum** conditionne la validité d'un modèle. Plus il est élevé, plus l'extension d'un modèle voué à être éliminé va s'interrompre tôt car il ne respectera plus ce quorum. Augmenter le quorum améliore donc les performances de l'extraction, et diminue le nombre de modèles produits. En contrepartie, les modèles qui ont une faible couverture sont éliminés ;
- **la longueur minimale** des modèles est le seuil à partir duquel les modèles valides sont reportés comme respectant les critères de l'extraction. Cependant, la longueur minimale n'a pas d'incidence sur le parcours dans les arbres des modèles et des suffixes : à critères identiques (dont la longueur maximale) mais avec deux longueurs minimales différentes, deux extractions se dérouleront strictement de la même manière, seuls les modèles produits différeront. L'utilisation des extensions maximales permet en théorie de profiter de cette différence des longueurs minimales pour interrompre prématurément des extensions vouées à l'échec. Avec les extensions maximales, l'augmentation de la longueur minimale devrait accélérer les performances de l'algorithme ;
- **la longueur maximale** influe sur la durée d'extension des modèles. Plus elle augmente, plus les modèles risquent d'être étendus longtemps. Le nombre total de modèles reportés augmente éventuellement avec la longueur maximale, mais rappelons que le nombre de modèles valides ayant une longueur donnée diminue nécessairement quand cette longueur augmente. Par conséquent, l'augmentation de la longueur maximale n'a pas une incidence importante sur l'extraction ;
- **le nombre maximum de substitutions** agit sur le cardinal du voisinage d'un modèle : plus il augmente, plus ce voisinage est grand et donc plus le nombre d'occurrences que peut recueillir un modèle augmente. Sachant que cette expansion est exponentielle et que l'extension d'un modèle nécessite l'extension de chacune de ses occurrences, on se doute que l'influence de ce paramètre sur les performances de l'algorithme est considérable. En pratique, on constate cependant que SMILE est capable d'extraire des modèles assez dégénérés en des temps raisonnables (en fonction bien sûr de la taille du jeu de données) ;
- **la fourchette de saut** d'un intervalle influe de deux façons sur l'extraction. La longueur de saut agit sur le temps d'extension des occurrences du modèle en cours. En effet, durant le saut (dans les deux algorithmes que nous avons développés), chaque occurrence du modèle en cours est étendue par tous les chemins de l'arbre des suffixes. Par conséquent, un saut nécessite le parcours (sans comparaisons donc peu coûteux) d'un sous-arbre dont la taille croit avec la longueur du saut. En pratique, nous verrons qu'elle a une très faible incidence.
  La largeur de l'intervalle de saut agit sur le nombre d'occurrences récoltées au terme d'un saut. Ce nombre influe lui-même sur le coût de l'extension de la boîte qui suit le saut effectué ;
- **le delta** a, en théorie comme en pratique, un coût multiplicatif : plusieurs extractions sur plusieurs intervalles contigüs ont le même coût que l'extraction équivalente avec des deltas (avec notre implantation). Le delta n'a donc qu'un caractère pratique, évitant de multiples extractions, et n'apporte pas de surcoût dans l'absolu.

6.1.2.2 Sortie

L'extraction se termine par la création d'un fichier contenant l'ensemble des modèles respectant les critères requis, ainsi que le nombre de séquences dans lesquelles ils apparaissent (couverture en séquences) et le nombre d'occurrences qu'ils possèdent dans l'ensemble des séquences (couverture en occurrences). Optionnellement, la liste des positions et séquences de chacune des occurrences du modèle est également affichée.

Dans le cas de modèles structurés, le symbole '_' exprime le saut entre deux blocs, et chaque occurrence éventuellement affichée est suivie des intervalles de saut entre chacun de ses blocs.

Les modèles apparaissent dans leur ordre d'obtention, c'est-à-dire l'ordre lexicographique. À titre d'exemple, l'extraction d'un modèle composé de deux boîtes séparées par un *spacer* de longueur $[12; 16]$ donne, avec l'affichage des occurrences, la sortie suivante :

```
AGAACT_AAAAC    2
Seq  21     Pos  715     13
Seq 143     Pos 2012     14
Seq  21     Pos   78     16
3

AGACAC_CGATAA   2
Seq 107     Pos  883     12
Seq  66     Pos   43     12
2
...
```

Dans cet exemple, le modèle `AGAACT_AAAAC` apparaît dans deux séquences, et possède trois occurrences. La première occurrence de ce modèle se trouve dans la séquence 21, à la position 715, et ses deux boîtes sont séparées par un *spacer* de longueur 13.

Si on utilise les deltas, l'extraction génère autant de fichiers de sortie qu'il y a de combinaisons d'intervalles possibles, afin de distinguer les modèles produits en fonction de l'intervalle pris en compte.

Afin de rendre cette sortie plus lisible, nous avons écrit un petit programme de post-traitement, prenant le type de fichier précédent en entrée, et produisant l'alignement de toutes les occurrences de chaque modèle. Les substitutions y sont mises en évidence, permettant de remarquer aisément les positions les moins significatives. Pour les deux modèles précédents, on obtient par exemple :

```
(2/225 sequences)   AGAACT      [12-16]     AAAAC
                    ==============================
Seq 21   Pos 715    AGAACT actcaacagaaat    AAAtC
Seq 143  Pos 2012   AGcACT acgcggggtcttac   AAAAC
Seq 21   Pos 78     AGAACT catgtccaagttactt AAAcC
===> 3 occurrences

(2/225 sequences)   AGACAC      [12-16]     CGATAA
                    ===============================
Seq 107  Pos 883    AGACAg ctatcttgactg     CGATAA
Seq 66   Pos 43     AGAtAC ccgcactgaaca     CGATAA
===> 2 occurrences
```

6.1.3 Évaluation

Le programme décrit précédemment permet d'extraire d'un jeu de séquences les modèles respectant des critères structurels. Dans une démarche d'inférence de sites de fixation dont on suppose qu'ils possèdent certaines propriétés fonctionnelles, et donc un caractère exceptionnel vis-à-vis du reste de la séquence, il nous faut appliquer des critères de signification afin de tenter de distinguer ceux des modèles produits qui pourraient avoir une réelle fonction biologique.

L'évaluation de la signification des motifs fait partie intégrante des algorithmes d'optimisation, où elle sert à éliminer des motifs pas ou peu significatifs. Dans notre cas, elle vient en post-traitement des données extraites, et permet d'obtenir l'estimation d'une signification pour tous les modèles issus de l'extraction. On disposera ainsi d'une liste de modèles triés selon les résultats de leur évaluation au terme de cette étape.

Comme nous l'avons expliqué dans la section 3.3 page 63, il existe de nombreuses approches permettant d'évaluer le caractère exceptionnel d'un motif, c'est-à-dire de prédire si sa présence en un certain nombre d'exemplaires dans un jeu de séquences est due au hasard, c'est-à-dire à la composition de la séquence, ou à son éventuelle fonction biologique.

À l'époque où nous avons produit la première version de SMILE, aucune méthode d'évaluation *a priori*, par des formules de probabilités d'apparition, ne nous satisfaisait pleinement. La première raison était que nous utilisions des modèles structurés, ce qu'aucune méthode ne traitait à l'époque. La seconde est que nos modèles permettaient des substitutions en nombre quelconque (et éventuellement nul), et que les méthodes disponibles qui prenaient en compte de tels modèles ([Régnier and Szpankowski, 1997], [Reinert and Schbath, 1998], [Tompa, 1999]) étaient trop gourmandes en calculs. Enfin parce que l'estimation du nombre de séquences contenant au moins une occurrence d'un modèle (au lieu du nombre total d'occurrences d'un modèle) apporte des complications supplémentaires (Tompa traite ce type de couverture pour des motifs de petite taille uniquement).

Pour ces raisons, nous avons choisi d'implanter et d'utiliser une méthode de *simulation* par *shuffling* des séquences (voir [Karlin et al., 1989]) pour évaluer nos modèles structurés. Nous avons également implanté une méthode dite « contre un autre jeu de séquences », applicable dans certains cas. Ces deux approches fournissent des statistiques d'apparition qui permettent d'obtenir des probabilités « attendues » et d'utiliser les tests de z-score et du χ^2 décrits dans la section 3.3 pour évaluer les modèles.

Plus récemment, l'intérêt porté par plusieurs équipes de mathématiciens à l'évaluation des modèles structurés que nous manipulons nous a amenés à travailler ensemble afin d'intégrer dans SMILE plusieurs nouvelles méthodes d'évaluation de la probabilité d'apparition d'un modèle.

Il est important de garder à l'esprit que nos travaux se sont concentrés sur la conception de structures et d'algorithmes permettant d'extraire, entre autres, des sites de fixation sur des critères structurels. Les méthodes d'évaluation appliquées en aval de SMILE sont complètement indépendantes de ces travaux. On peut donc à loisir traiter les résultats d'une extraction avec n'importe quel outil d'évaluation qui peut gérer des modèles structurés. Les méthodes d'évaluation que nous avons intégrées à ce jour peuvent donc être facilement remplacées par d'autres méthodes plus efficaces si nécessaire.

6.1.3.1 Simulation par *shuffling* des données

Nous décrivions dans la section 3.3 la principale difficulté qui se pose quand on souhaite évaluer le caractère exceptionnel d'un motif : évaluer sa probabilité d'apparition, donc celle de ses occurrences. Une fois munis de cette probabilité, plusieurs tests permettent de calculer une mesure de ce caractère exceptionnel : z-score, test du χ^2,... Dans notre cas, la difficulté consiste donc à produire cette probabilité (et éventuellement un écart-type associé) de façon fiable pour des modèles structurés.

La première solution que nous avons retenue est la méthode par simulation. Elle consiste à générer un jeu de N séquences aléatoires dont la $i^{\text{ème}}$ possède la même composition en k-mers que la $i^{\text{ème}}$ des N séquences originales. On appelle ces séquences des séquences *shufflées* et le processus de mélange *shuffling*. On recherche [2] ensuite les modèles obtenus lors de l'extraction dans ces séquences shufflées, et on conserve les couvertures en séquences obtenues pour chaque modèle. On répète ces opérations de *shuffling*/recherche un certain nombre de fois afin de stabiliser la moyenne des résultats obtenus. Au terme du processus, cette moyenne est considérée comme la couverture du modèle due à la composition de la séquence. Si le modèle a une fonction biologique, on estime qu'il devrait apparaître plus fréquemment que ce qu'on observe dans les séquences shufflées.

Afin de mesurer ce caractère exceptionnel, on applique le test du z-score. Conformément

[2] nous avons développé pour cette recherche un outil nommé **grep+**, qui réalise la recherche dans un jeu de séquences des modèles structurés ayant un nombre quelconque de boîtes, avec substitutions, autorisant les alphabets dégénérés et les expressions régulières simples. Cet outil a été intégré dans l'interface d'un logiciel d'analyse de données biologiques développé à l'institut Pasteur par Catherine Letondal, dénommé **Biok** (`http ://www-alt.pasteur.fr/~letondal/biok/`).

à la définition donnée dans la section 3.3 page 63, nous utilisons donc la formule suivante pour un modèle m :

$$\text{z-score}(m) = \frac{P_{seq} - P_{shuf}}{\sqrt{\sigma}}$$

où P_{seq} est le nombre de séquences du jeu original dans lesquelles une occurrence du modèle m apparaît au moins une fois, P_{shuf} son équivalent dans les séquences shufflées, et σ l'écart-type calculé à partir des distributions observées dans les séquences shufflées.

On applique également le test du χ^2 à un degré de liberté sur les résultats de la simulation. La formule utilisée est également celle de la section 3.3 :

$$\chi^2(m) = 2\left(\frac{P_{seq}.A_{shuf} - P_{shuf}.A_{seq}}{n_P.n_A.nbseq}\right)^2$$

où $nbseq$ est le nombre de séquences (originales ou shufflées, c'est le même nombre), A_{seq} le nombre de séquences originales dans lesquelles aucune occurrence du modèle m n'apparaît, A_{shuf} son équivalent pour les séquences shufflées, $n_P = P_{seq} + P_{shuf}$ et $n_A = A_{seq} + A_{shuf}$.

Au terme de la simulation, on dispose donc de deux mesures de la signification statistique des modèles obtenus lors de l'extraction. Le χ^2 ne permettant pas de distinguer le jeu sur-représenté (la valeur produite indique juste l'écart de cohérence entre les deux résultats) et n'étant pas normalisé, on se base sur le z-score pour trier les résultats : du plus grand (le modèle est sur-représenté par rapport à ce qu'on a obtenu par simulation) au plus petit (le modèle est sous-représenté). Les valeurs de χ^2 viennent confirmer ou altérer le classement obtenu grâce au z-score.

La méthode de simulation apporte de grands avantages et de grands inconvénients. Côté avantages, elle permet d'obtenir une probabilité pour des modèles complexes, qui est délicate à évaluer par des formules explicites de probabilités. De plus, les probabilités obtenues semblent être assez fiables quand un nombre suffisant de *shufflings*/recherches est réalisé. Côté inconvénients, elle est assez coûteuse en calculs. La stabilisation des résultats nécessite un nombre important de *shufflings* (nous validons nos résultats avec mille *shufflings* en général). Afin de minimiser le coût de la recherche d'un modèle dans les séquences shufflées, nous avons à nouveau utilisé un arbre des suffixes pour rechercher les modèles dans ces séquences. Malgré cela, une recherche de modèle structuré reste coûteuse, et le processus complet d'évaluation du grand nombre de modèles éventuellement recueilli lors de l'extraction peut demander de gros temps de calcul [3].

L'évaluation est intégrée dans SMILE par le biais du fichier de paramètres décrit précédemment. Il suffit d'y ajouter les lignes suivantes :

[3] le *shuffling*, le mélange des séquences, n'est pas en lui-même très coûteux. Pour l'implanter, nous avons récupéré et adapté le programme **Shufflet** d'Eivind Coward ([Coward, 1999]).

```
EVALUATION (Step 2) ============================
Shufflings                     100
Size k-mer                     2
```

La première ligne de paramètres indique le nombre de *shufflings* à réaliser lors de la simulation, la seconde la longueur des mots à conserver lors de ces *shufflings*. On conserve en général les di- ou tri-nucléotides. Des tests présentés en annexe font état de la variation des résultats observés en fonction de la longueur des k-mers conservés lors des *shufflings*.

Ajoutée de la sorte dans le fichier de paramètres, la simulation est automatiquement enchaînée après l'extraction, avec les modèles obtenus. Un fichier est produit, qui contient les résultats de cette évaluation. Il se présente sous la forme suivante :

```
STATISTICS ON THE NUMBER OF SEQUENCES HAVING AT LEAST ONE OCCURRENCE
Model              %right #right  %shfl.  #shfl.   Var.    Chi2  Z-score
========================================================================
TTGCGC_TATAATGC    2.27%      10   0.01%    0.03   0.18    10.02   56.61
TTGCGC_ATAATGC     2.04%       9   0.03%    0.12   0.35    8.73    25.50
TCTTGA_GCTATA      2.04%       9   0.04%    0.18   0.41    8.55    21.28
TGCTTG_GTATAA      2.49%      11   0.06%    0.27   0.53    10.35   20.30
CTTGAA_GCTATA      2.04%       9   0.05%    0.20   0.45    8.50    19.41
GCTTGA_GTATAA      2.04%       9   0.05%    0.22   0.47    8.46    18.69
CTTGAC_TATAAT      2.27%      10   0.07%    0.29   0.52    9.28    18.53
TTGACT_ATAATG      2.49%      11   0.09%    0.38   0.61    10.05   17.48
TTCACA_CGATAA      2.04%       9   0.06%    0.27   0.52    8.32    16.82
GCTTGA_TATAAT      2.27%      10   0.08%    0.34   0.58    9.14    16.64
TTGACT_TAAAAT      2.95%      13   0.14%    0.60   0.80    11.48   15.57
TTTGTT_CTTGTG      2.49%      11   0.11%    0.48   0.68    9.76    15.47
...
```

Ce sont les résultats obtenus en utilisant les couvertures en séquences des modèles. À gauche figure le modèle concerné, puis le pourcentage de séquences originales où il apparaît, leur nombre, le pourcentage de séquences shufflées où il apparaît, leur nombre moyen sur l'ensemble des *shufflings*, la variance obtenue, et enfin les résultats des tests de χ^2 et z-score.

Dans cet exemple, tiré d'une application que nous traiterons dans le chapitre suivant, le premier modèle se détache nettement par son z-score, tandis que le χ^2 met en avant un autre modèle. Cette différence peut résulter d'un nombre insuffisant de simulations. Cependant, si comme ici des différences locales apparaissent souvent dans les classements obtenus grâce aux deux tests, le classement global est en général le même : les dix modèles ayant les meilleurs z-scores ont souvent les dix meilleurs χ^2. Les deux tests montrent en général une bonne corrélation entre eux.

Toujours dans le fichier produit par la simulation suivent les résultats obtenus en utilisant les couvertures en nombre d'occurrences des modèles :

```
STATISTICS ON THE TOTAL NUMBER OF OCCURRENCES
Model               #right    #shfl. Var.    Chi2    Z-score
============================================================
TTGCGC_TATAATGC       10      0.03   0.18    9.91    56.61
TTGCGC_ATAATGC         9      0.12   0.35    8.64    25.50
TCTTGA_GCTATA          9      0.18   0.42    8.46    21.12
TGCTTG_GTATAA         11      0.27   0.53    10.22   20.30
CTTGAA_GCTATA          9      0.20   0.46    8.41    19.29
GCTTGA_GTATAA          9      0.22   0.47    8.38    18.69
CTTGAC_TATAAT         10      0.29   0.52    9.17    18.53
TTGACT_ATAATG         11      0.38   0.61    9.92    17.48
TTCACA_CGATAA          9      0.27   0.52    8.23    16.82
GCTTGA_TATAAT         10      0.34   0.58    9.04    16.64
GTTGTC_TATAAT          9      0.33   0.57    8.06    15.12
TTGACT_TAAAAT         13      0.61   0.82    11.28   15.11
...
```

Les colonnes contiennent les mêmes indications que précédemment, excepté que les pourcentages (qui n'ont pas grand sens ici) ont disparu.

Dans cet exemple, le classement des z-scores met en avant le même modèle que précédemment. Le modèle `TTGCGC_TATAATGC` semble donc être exceptionnellement présent dans le jeu de séquences utilisées. Les résultats obtenus selon les deux modes de couverture considérés sont en général très proches.

6.1.3.2 Contre un autre jeu de séquences

La seconde méthode employée est moins neutre que la première : il faut en effet disposer d'un autre jeu de séquences auquel le jeu original puisse se comparer. On utilisera par exemple un jeu de séquences dont on sait qu'elles ne contiennent pas le site de fixation qu'on suppose se trouver dans le jeu original. Notre problème de conservation s'apparente alors un peu plus à un problème de classification (voir section 3.2 page 59).

Seul le test du χ^2 peut être employé dans ce cas, puisqu'on ne peut calculer d'écart-type avec les seules couvertures relevées dans le second jeu de séquences.

La sortie obtenue est proche de celle de la simulation, un signe indique si le modèle concerné est sur- ou sous-représenté (puisque la valeur du χ^2 seule ne suffit pas pour statuer). Les plus forts χ^2 indiquent les modèles mieux ou moins bien (en fonction du signe associé) représentés dans le jeu original qu'ils ne le devraient en se basant sur le second jeu de séquences. Un χ^2 nul indique que le modèle est représenté de la même manière dans les deux jeux.

La composition des séquences du jeu original et du second jeu n'entrent pas en compte dans la mesure utilisée. Cela peut d'ailleurs poser un problème lorsque les biais de composition diffèrent fortement entre les deux jeux.

Cette méthode d'évaluation de la signification statistique est très rapide, puisqu'il suffit de rechercher les modèles obtenus lors de l'extraction dans le second jeu. Il faut cependant disposer d'un tel jeu de séquences, ce qui est rarement le cas.

6.1.3.3 Probabilités d'apparition d'un modèle

Deux équipes de mathématiciens se sont intéressées au problème de l'évaluation de la probabilité d'apparition des modèles structurés que nous avons développés :
- Mireille Régnier, de l'INRIA Rocquencourt, et ses étudiants : Jitesh Jain et Mathias Vandenbogaert ;
- Jean-Jacques Daudin, Stéphane Robin, et Sophie Schbath, de l'INA P-G, ainsi qu'un de leurs étudiants, Hugues Richard.

Elles ont développé deux méthodes différentes qui permettent d'estimer la signification des modèles structurés (réduits à deux boîtes pour le moment). Ces méthodes sont décrites dans [Régnier and Szpankowski, 1997] et [Robin et al., 2001]. À ce jour, nous n'avons pas encore intégré ces méthodes d'évaluation dans la seconde partie de SMILE, leurs outils et formules étant toujours en cours de validation. Nous disposons cependant de résultats faisant état d'une bonne corrélation avec ceux obtenus par simulation, et ce malgré le fait que leurs probabilités d'apparition soient actuellement calculées par des approximations.

Étant donné le coût élevé de la simulation (nous verrons dans le chapitre suivant que c'est en général cette étape qui grève lourdement les temps de calcul d'une inférence réalisée avec SMILE), il semblerait que ces approches puissent s'y substituer très avantageusement. L'intégration de ces méthodes d'évaluation est donc un travail à effectuer en priorité.

6.2 Exemples de résultats et performances

En dehors de toute considération biologique, nous présentons dans cette section des exemples de résultats et performances des algorithmes d'extraction développés pour SMILE. Les extractions effectuées s'appliquent à des modèles simples et structurés. Nous testons également SMILE sur un problème présenté récemment dans [Pevzner and Sze, 2000] comme un « *challenge* » en matière d'extraction, et auquel se réfèrent plusieurs auteurs d'algorithmes récents ([Pavesi et al., 2001], [Eskin and Pevzner, 2001], [Sze et al., 2002]).

Tous les tests ont été réalisés à l'IGM sur un PC bi-Pentium III 1Ghz avec 512Mo de mémoire vive, avec la version 1.43 de SMILE. Les performances sont mesurées en temps CPU, c'est-à-dire le temps que le processeur a passé à traiter le processus.

6.2.1 L'arbre des facteurs

Le test réalisé ici utilise un jeu de 2000 séquences de longueur 500 (pour un poids total de 1Mbases), générées aléatoirement sur l'alphabet A, C, G, T avec équiprobabilité des symboles. Il nous montre l'influence de la longueur maximum des facteurs sur la taille de l'arbre. Les temps de construction, très courts, n'ont pas été étudiés (se reporter à [Allali, 2000] pour plus de détails). La figure 6.2 montre les résultats obtenus.

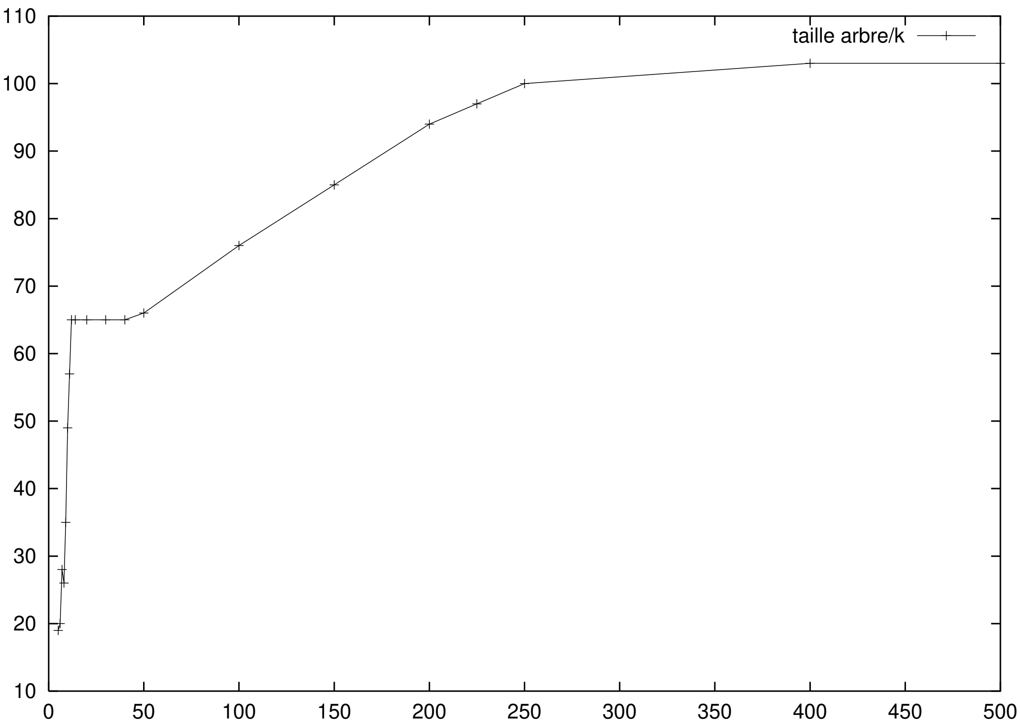

Fig. 6.2 – Taille de l'arbre des facteurs en fonction de la longueur maximum des modèles.
La taille de l'arbre est exprimée en Mo. Le jeu de données contient 2000 séquences de 500 symboles chacunes.

Ces résultats confirment que l'utilisation d'un tel arbre peut réduire considérablement l'espace mémoire nécessaire. On conclut également que, moyennant une limitation de la longueur maximum des modèles, SMILE est utilisable pour de gros jeux de données sur des machines n'ayant pas nécessairement une gigantesque quantité de mémoire vive. En effet, la majeure partie de la mémoire requise par SMILE provient de la structure de données ; après sa construction et le lancement de l'extraction, le programme est généralement très stable en mémoire (cela dépend de la façon dont se déroule l'extraction, et plus précisément de la quantité d'occurrences empilées pendant celle-ci).

6.2.2 Modèles simples

Les tests réalisés dans cette section utilisent le jeu précédent de 2000 séquences de 500 symboles, contenant 1Mbases. Comme observé précédemment, l'arbre des suffixes généré pour ce jeu requiert 103Mo de mémoire pour une extraction sans limite de longueur. En limitant la longueur maximum des motifs à 20, la mémoire requise est réduite à 65Mo.

Nos tests portent sur les temps nécessaires à l'extraction de différents types de modèles simples, avec des critères variés. Le tableau suivant reporte les résultats obtenus :

quorum	nb subst.	long. mod.	alphabet	nb mod.	temps
20%	1	$[8;+\infty[$	A,C,G,T	10	2.5 sec.
50%	2	$[8;+\infty[$	A,C,G,T	69 338	2 mn. 17 sec.
100%	2	$[8;+\infty[$	A,C,G,T	0	13 sec.
100%	1	$[8;+\infty[$	A,C,G,T,R,Y	5 591	4 mn. 30 sec.
3%	1	$[10;+\infty[$	A,C,G,T	6	21 sec.
10%	2	$[10;+\infty[$	A,C,G,T	1 048 575	9 mn. 7 sec.

Ces tests donnent un ordre de grandeur pour certaines extractions. On note que ce n'est pas tant le nombre de substitutions que le nombre de modèles visités et recueillis qui accroît le temps d'exécution (ces deux nombres étant évidemment fortement liés). Le dernier test est visiblement trop peu contraint, puisque, même si certains modèles recueillis peuvent avoir une longueur supérieure à 10, le nombre de modèles possibles de cette longueur est de l'ordre de 2 millions et on en a recueilli la moitié. Ce filtrage n'est par conséquent pas très intéressant.

Les deux tests suivants ont pour but d'observer les performances de SMILE lors de l'extraction d'un signal inséré artificiellement et ayant une faible couverture. Nous générons un jeu de 500 séquences de 1000 bases (pour un total de 500Kbases) de la même manière que précédemment, dans lequel nous insérons un modèle. Nous essayons ensuite d'extraire ce modèle avec des critères adaptés.

Lors du premier test, nous avons inséré un modèle de longueur 11 avec 2 substitutions dans 20% des séquences. Dans le second test, le modèle a également une longueur 11 mais une seule substitution et deux positions recevant une purine, dans 10% des séquences. Les résultats figurent dans le tableau suivant :

quorum	nb subst.	long. mod.	alphabet	nb mod.	temps
20%	2	$[11;+\infty[$	A,C,G,T	3	7 mn. 32 sec.
10%	1	$[11;+\infty[$	A,C,G,T,R	1	4 mn. 30 sec.

Utilisant des quorum bas, ces tests montrent la capacité de SMILE à dégager un modèle à faible couverture du bruit de la séquence. Cette situation (quorum bas, jeu de données conséquent) correspond à celles que nous avons souvent rencontrées dans certaines applications. En effet, lorsqu'on dispose d'un génome annoté, sans autre information supplémentaire sur des gènes qui partageraient les mêmes familles de promoteurs ou de sites de régulation, on doit, pour détecter ces sites, pratiquer une extraction sur l'ensemble des séquences non codantes se trouvant en amont des gènes (ou du moins des portions de ces séquences précédant les débuts de traduction annotés). La taille du jeu de données peut par conséquent être assez grande, voire gigantesque chez des organismes supérieurs.

Ces tests mettent en avant une capacité à traiter en des temps très raisonnables de grandes quantités de données. Ces bonnes performances sont d'importance : SMILE étant un outil qui réalise des inférences sur des critères structurels, il nécessite d'être utilisé avec certains paramètres. Dans le cas où les caractéristiques d'un site recherché sont mal connues, il est nécessaire de procéder par « tâtonnements », et donc de disposer d'un outil performant.

6.2.3 Le « *challenge* » de Pevzner et Sze

Dans une publication de Pevzner et Sze parue récemment ([Pevzner and Sze, 2000]) est défini, au sujet du problème de l'inférence de sites de fixation, un « *challenge problem* » qu'aucun outil disponible n'est capable de résoudre correctement. Les auteurs présentent un algorithme nouveau de type combinatoire heuristique qui traite le problème défini.

Nous avons discuté au terme du chapitre 3 de la pertinence discutable, à nos yeux, de ce *challenge* en tant que problème représentatif des problèmes d'inférence de sites de fixation. D'un pur point de vue algorithmique, nous avons souhaité expérimenter SMILE sur ce « challenge » afin de montrer qu'il peut être résolu efficacement de façon exacte.

Le problème posé consiste à détecter un motif de longueur 15 avec 4 substitutions disséminé parmi 20 séquences de longueur 100 à 4000. L'heuristique proposée par Pevzner et Sze affiche une bonne stabilité des résultats avec 88% à 98% de réussite (pour le meilleur algorithme : WINNOWER avec k=3) selon la longueur des séquences, mais les temps d'exécution sont dits être excessifs sans plus amples détails. Les complexités pour cet algorithme affichent un facteur $n^{2.66}$ (où n est la taille des séquences du jeu) qui le rendent difficilement applicable sur des jeux de données tels ceux que nous avons utilisés précédemment.

Sur des jeux de données aléatoires similaires, contenant un modèle de longueur 15 avec 4 substitutions, SMILE affiche les résultats suivants (figure 6.3) :

Longueur seq.	100	200	300	400	500	600	700	800	900
Temps CPU (mn)	0.59	2.30	4.53	7.57	11.26	15.37	20.01	25.50	31.97

Longueur seq.	1000	2000	4000
Temps CPU (mn)	37.58	123.33	360

SMILE obtient des résultats intéressants sur ce problème, compte tenu du fait que l'algorithme est exact, et que le modèle inséré est donc retrouvé systématiquement. De plus, des tests d'optimisation par les extensions maximales similaires à ceux effectués dans le chapitre précédent révèlent que, lors de l'extraction sur le jeu de 20 séquences de longueur 600, 53.12% des visites de modèles peuvent être évitées dans le meilleur des cas. L'implantation des extensions maximales devrait donc améliorer notablement les extractions appliquées à ce « *challenge* ».

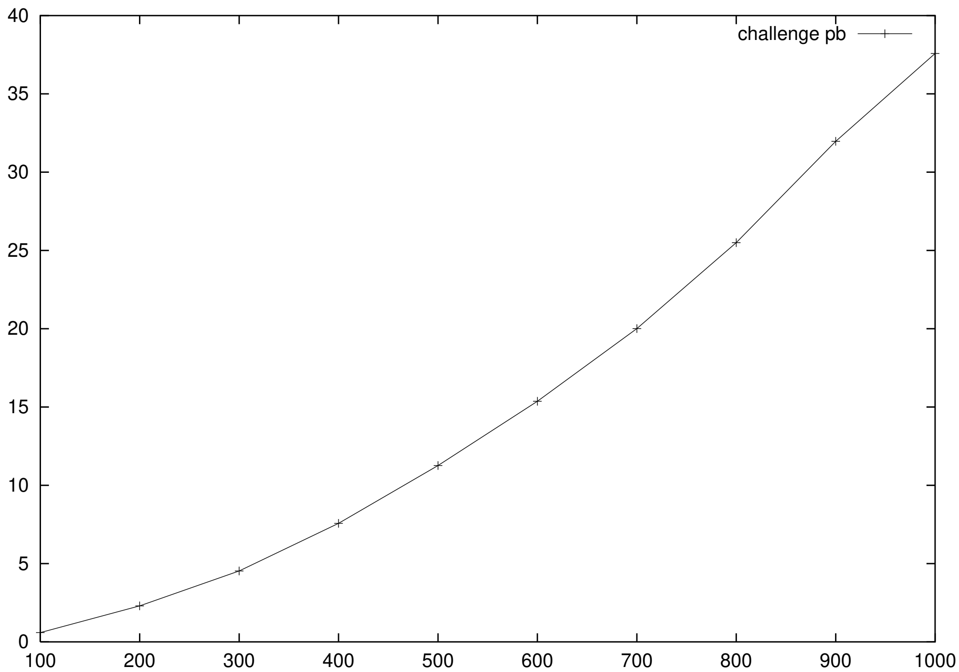

FIG. 6.3 – Temps d'exécution de SMILE pour le « *challenge problem* ».
En abscisse, le nombre de symboles dans chacune des vingt séquences, et en ordonnée le temps en minutes nécessaire à SMILE pour extraire le signal de longueur 15 avec 4 substitutions.

Le problème posé par Pevzner et Sze ne nous semble cependant pas pertinent pour évaluer des algorithmes destinés à détecter des sites de fixation dans des séquences génomiques. Cela pour deux raisons :

- il concerne de petits jeux de séquences (de 2Kb à 20Kb), tandis que, comme expliqué dans la section précédente, les jeux de données dans lesquels nous sommes susceptibles de chercher à inférer des sites de fixation sont généralement bien plus larges, même quand on dispose de données d'expression utilisables pour isoler des gènes pouvant partager des promoteurs communs ;
- dans la seule application biologique illustrant cet article, le site extrait est présenté comme étant « à la limite de ce qui est détectable » pour les outils d'inférence existants. Or ce site ($TGTGAnnnngnTCACA$) est perçu comme un long motif très dégénéré, alors qu'il contient un *spacer*. Nous estimons que la pertinence des *spacers* pour représenter plusieurs sites de fixation et la combinatoire engendrée par ces *spacers* sont meilleures que lorsqu'on se contente d'augmenter le nombre de substitutions d'un unique motif long.

Les difficultés rencontrées lors de l'inférence de sites de fixation résident, selon nous, pour une grande part dans l'utilisation de représentations inadaptées pour modéliser les signaux recherchés. Le « *challenge* » présenté par Pevzner *et al.* palie à cette faiblesse de représentation en augmentant considérablement l'espace des solutions, et a donc recours à une heuristique pour le résoudre. Notre choix d'exprimer des *spacers* de longueur variable nous semble être plus pertinent biologiquement, et moins coûteux d'un point de vue algorithmique, ce qui nous permet de résoudre le problème de façon exacte.

6.2.4 Modèles structurés

Nous expérimentons maintenant l'extraction de modèles structurés d'un jeu de 200 séquences de 1000 bases (soit 200Kbases) générées aléatoirement sur un alphabet A, C, G, T. Nous avons inséré dans une portion de ces séquences un modèle composé de deux boîtes de longueur 6 séparées par 8 à 10 symboles.

Toutes les extractions ont été réalisées pour deux boîtes de longueur minimum 6 et sans longueur maximum. Dans un premier temps, nous cherchons à extraire un modèle structuré exact disséminé dans 10% des séquences :

susbt. glob.	subst. boîtes	intervalle saut	delta	nb mod.	temps
0	0 et 0	$[8-10]$	Non	1	1.0 sec.
0	0 et 0	$[4-14]$	1	1	7.7 sec.

Sans substitutions, les extractions s'avèrent très rapides.

Les tests suivants visent à montrer l'influence de la largeur de l'intervalle de saut, de la longueur des sauts effectués et de l'utilisation des deltas. Nous altérons le signal d'une substitution dans une des deux boîtes.

susbt. glob.	subst. boîtes	intervalle saut	delta	nb mod.	temps
1	1 et 1	$[8-10]$	Non	1	2.33 mn.
1	1 et 1	$[7-12]$	Non	1	5.68 mn.
1	1 et 1	$[3-14]$	Non	7	12.52 mn.
1	1 et 1	$[0-23]$	Non	24930	29.34 mn.
1	1 et 1	$[7-12]$	Non	1	5.68 mn.
1	1 et 1	$[7-12]$	1	1	9.77 mn.
1	1 et 1	$[8-10]$	Non	1	2.33 mn.
1	1 et 1	$[14-16]$	Non	1	2.32 mn.
1	1 et 1	$[50-52]$	Non	1	2.18 mn.

Le premier test est celui dont les critères sont les plus proches du signal inséré. Dans la première série de tests comportant quatre extractions, la largeur de l'intervalle de saut vaut 3, 6, puis 12 et 24. Elle est donc multipliée par deux à chaque fois, et on observe que les temps d'exécution croissent linéairement. L'implantation que nous avons effectuée semble donc être linéaire en la largeur de l'intervalle de saut.

La seconde partie des tests montre l'influence de l'utilisation du delta. On note que le temps réalisé pour l'extraction avec $\delta = 1$ sur l'intervalle $[7-12]$ (visitant par conséquent quatre intervalles dans cette fourchette) correspond à peu près au coût de quatre extractions sur l'intervalle $[8-10]$ (dont la largeur vaut 1). Le fait d'utiliser les deltas devrait théoriquement permettre de « factoriser » certaines extensions, et par conséquent de rendre une extraction à l'aide des deltas moins coûteuse que plusieurs extractions sans delta équivalentes. Cependant, notre implantation privilégie actuellement l'espace sur le temps. Les deltas ont donc essentiellement un aspect pratique pour le moment.

La dernière partie montre l'influence de l'augmentation de la longueur du *spacer* sur les temps de calcul. Dans ces trois tests, nous avons étendu le *spacer* initial de $[8-10]$ qui compose le signal original aux intervalles $[14-16]$ et $[50-52]$. On observe que cette extension n'apporte pas de surcoût à l'extraction. À notre grande surprise, elle diminue même les temps d'extraction, ce que nous n'expliquons pas. Ces résultats confirment le fait que la descente sans comparaisons dans l'arbre des suffixes est beaucoup moins coûteuse que la descente classique. Ils nous confortent en tout cas dans l'idée que la représentation des modèles structurés que nous avons développée s'avère algorithmiquement intéressante pour modéliser des sites de fixation séparés par des *spacers* éventuellement très longs, ce qui est pertinent d'un point de vue biologique puisqu'on observe parfois des interactions à longue (voir très longue) distance entre sites de fixation.

Pour finir, si les extractions sont notablement ralenties par l'augmentation du nombre de substitutions, les temps obtenus sont raisonnables si l'on considère la complexité du modèle extrait, la faible couverture de ce dernier et la taille du jeu de données.

Nous altérons cette fois le signal original d'une substitution dans chacune des deux boîtes et répartissons le modèle dans 15% des séquences :

susbt. glob.	subst. boîtes	intervalle saut	delta	nb mod.	temps
2	1 et 1	$[8-10]$	Non	55	19.34 mn.

L'influence du nombre de substitutions semble, comme prévu, importante.

Enfin, afin d'observer l'influence du nombre de boîtes sur le fonctionnement de l'algorithme, nous plaçons cette fois dans les séquences aléatoires un signal composé de trois boîtes de longueur 4, avec une substitution globale et deux intervalles de sauts de $[6-8]$ et $[12-14]$. Ce signal a donc la même longueur (12) que le signal original, mais elle est répartie sur trois boîtes au lieu de deux.

susbt. glob.	subst. boîtes	intervalles saut	delta	nb mod.	temps
1	1, 1 et 1	$[6-8], [12-14]$	Non	1	11.98 mn.

Notons que la combinatoire des modèles structurés peut augmenter considérablement avec le nombre de boîtes, en particulier si on utilise des deltas sur les différents intervalles.

Les résultats obtenus nous paraissent concluants ; on peut en effet espérer être capable d'extraire des signaux relativement faibles en couverture, correctement conservés, et structurés en plusieurs boîtes, d'un jeu de séquences de taille raisonnable. Nous verrons dans le chapitre suivant que ces conditions correspondent bien à celles que nous avons rencontrées lors de certaines applications.

6.3 Améliorations possibles...

SMILE est complètement fonctionnel et utilisable, tous les critères décrits dans ce chapitre ont été intégrés. Il est distribué depuis le début de l'année 2001.

Cependant, faute de temps, plusieurs travaux d'implantation restent à faire à ce jour pour que ce logiciel soit plus efficace et plus aisément accessible :

- SMILE fonctionne actuellement par l'intermédiaire d'un fichier de paramètres. Il nous reste à développer une interface graphique qui le rende plus convivial, en particulier pour les biologistes qui peuvent être amenés à l'utiliser. Le développement d'une interface Web est aussi un point important et nécessaire à sa diffusion ;
- l'algorithme 2, d'extraction de modèles structurés par le biais d'un arbre dynamique, n'a pas été implanté. Ce travail, qui nécessite de gros changements dans les structures de données utilisées, devrait théoriquement rendre l'algorithme plus efficace pour extraire des modèles de deux boîtes et plus. Il nous reste à déterminer si les gains en pratique sont suffisants pour nécessiter une implantation ;
- les extensions maximales semblent apporter une optimisation importante aux algorithmes d'extraction moyennant une implantation relativement facile et économique. L'intégration de la gestion des extensions maximales dans le programme est vivement souhaitable. Elle permettrait dans un premier temps de vérifier si le surcoût en temps qu'elles apportent est compensé par l'optimisation apportée, et, si ces expériences sont concluantes, de rendre à terme le programme plus rapide dans l'étape d'extraction ;
- les nouvelles méthodes d'évaluation *a priori* décrites dans ce chapitre doivent aussi être intégrées parmi les outils servant à estimer la signification des modèles ;
- la gestion des deltas doit être modifiée de façon à diminuer leur coût.

D'autres idées, moins abouties, permettraient de simplifier l'utilisation de SMILE. Par exemple, parmi les critères obligatoires requis, le quorum est le plus contraignant. Il oblige en effet l'utilisateur à estimer la couverture d'un site dont il connaît parfois très mal les caractéristiques. Si la méthode du tâtonnement permet de cerner rapidement les quorums intéressants pour un jeu de séquences données, une des extensions majeures que nous souhaiterions apporter à nos algorithmes et à SMILE consisterait à s'affranchir de ce quorum. C'est envisageable en considérant que l'on peut réaliser le travail de tâtonnement à l'intérieur même de l'algorithme d'extraction. Il suffirait pour ce faire de déterminer à l'avance, en effectuant un premier parcours des séquences, les seuils de quorum au-delà desquels l'extraction n'est pas pertinente, et de réaliser en une passe l'extraction classifiée de tous les modèles ayant une couverture comprise dans une fourchette de quorum susceptible de présenter un intérêt.

Il serait également utile de remplacer le nombre de substitutions maximum par un taux dépendant de la longueur du motif. Cette modification assez aisée est cependant incompatible avec l'extraction de modèles de longueur illimitée (auquel cas le nombre de substitutions l'est aussi !). Il faudrait donc retirer cette possibilité en rendant la saisie d'une longueur maximum obligatoire. L'algorithme de Pavesi *et al.*, qui s'inspire de notre approche, prend pour parti de limiter le nombre de substitutions par tranche de motif : lors de l'extraction d'un motif ayant au plus 25% de substitutions, les quatre premières positions ne pourront par exemple pas subir plus d'une substitution ([Pavesi et al., 2001]). Cette contrainte, disqualifiant une partie de l'espace des solutions, fait de cet algorithme une heuristique.

6.4 Discussion

Le programme que nous avons développé se révèle être efficace pour extraire dans des temps raisonnables des signaux complexes artificiellement dissimulés dans des jeux de données aléatoires de taille conséquente. Si SMILE est fortement ralenti par l'augmentation du nombre de substitutions, l'usage des *spacers* s'avère être raisonnable en temps de calcul. L'utilisation des *spacers* pour représenter des sites de fixation, sensée améliorer sensibilité et spécificité à certains sites, devrait justement permettre d'éviter d'augmenter le nombre de substitutions nécessaires lors de leurs extractions.

Dans un cas réel, avec peu de connaissances du modèle à extraire, SMILE requiert de progresser en tâtonnant comme nous le verrons dans le chapitre qui suit, par extractions et évaluations successives, afin de cerner les caractéristiques structurelles les plus pertinentes. Les performances du programme sont donc d'importance.

Bien que SMILE nécessite un certain nombre de critères d'extraction, il permet d'approcher un problème dont on ne connaît pas nécessairement bien les caractéristiques grâce à la souplesse de certains d'entre eux : les longueurs des boîtes et des *spacers* peuvent être définies par des intervalles larges. Seul le quorum, qui est le critère filtrant le plus important, requiert un minimum de connaissances de la couverture du modèle recherché.

Pour l'utilisateur, SMILE diffère radicalement des outils d'inférence les plus couramment employés, tels MEME, Gibbs Sampler ou Motifs Sampler. L'interface de ces programmes d'optimisation ne requiert que peu de connaissances des modèles ; le principe d'optimisation permet de converger vers des sites qui ont un caractère significatif, en faisant varier leur longueur ou leur couverture. En contrepartie, le motif est décrit de façon moins fine que nous le faisons, rendant ces approches assez sensibles au bruit. En effet, les critères structurels, s'ils sont contraignants quand il s'agit d'inférer un site dont on connaît peu de caractéristiques, peuvent devenir pertinents lorsque la structure du site est un peu mieux connue (après tâtonnements ou par le biais de certaines connaissances *a priori*). Ils permettent de mieux décrire certains sites, de façon plus spécifique, donc de mieux les isoler du bruit. Nous verrons plus en détail dans le chapitre suivant les différences d'utilisation et de sensibilité des approches combinatoires et d'optimisation face à des séquences biologiques.

Après ces quelques tests, nous pouvons toutefois situer le domaine d'utilisation de SMILE parmi les outils existants : la nécessité de connaître un minimum de critères pour extraire un site inconnu peut rendre fastidieuse la recherche par tâtonnement des critères les mieux adaptés. À ce stade, les algorithmes d'optimisation, peu gourmands en critères, ont l'avantage de pouvoir parfois (pourvu que le site soit suffisamment représenté dans les séquences) dégager le patron d'un site qu'on pourra ensuite extraire de façon plus fine avec SMILE, en essayant éventuellement d'associer des modèles simples en modèles structurés, ce qu'aucun algorithme d'optimisation ne fait actuellement. Notre outil vient dans ce cas en complément des approches existantes.

SMILE offre aussi l'avantage de produire une classification de tous les modèles obtenus, là où ces mêmes algorithmes d'optimisation ne peuvent produire qu'un nombre limité de motifs. L'utilisateur a également un droit de regard sur la façon dont les candidats sont évalués après extraction, ce qui permet de comparer plusieurs méthodes d'évaluation.

Pour ces raisons, nous pensons que SMILE comble un vide parmi les outils dont disposent les bio-informaticiens pour extraire des signaux. MEME (qui est un des outils les plus utilisés actuellement dans ce but) est d'un fonctionnement tellement simple que l'on constate avec surprise que peu de ses utilisateurs sont conscients de ses limitations, du type de signaux qu'il peut et ne peut pas détecter. En comparaison, SMILE nécessite un réel apprentissage de son fonctionnement et une interaction de l'utilisateur avec le programme, en échange de la garantie que tous les modèles respectant les critères décrits seront extraits. SMILE requiert donc un investissement et une approche plus rigoureuse de l'inférence, qui a tendance à prendre malheureusement parfois des allures de recette magique aux yeux d'utilisateurs non avertis.

Chapitre 7

Applications

Disposant d'un outil complet et utilisable, notre objectif est maintenant d'évaluer la pertinence des représentations et algorithmes développés dans ce mémoire en appliquant SMILE à la détection de signaux dans des jeux de données d'origine biologique.

Dans un premier temps, les expériences que nous avons menées visaient à tester la capacité de SMILE à détecter des signaux connus dans des jeux de données propres, où ces signaux étaient fortement représentés. La section 7.1 décrit les résultats obtenus sur des séquences sélectionnées expérimentalement dans les génomes d'*Escherichia coli* et *Bacillus subtilis*.

Nous avons ensuite mené des expériences sur des jeux de données bruts issus des génomes annotés de *E.coli*, *B.subtilis* et *Helicobacter pylori*, et dont les résultats sont décrits dans la section 7.2.

Ces deux séries d'expériences visaient à identifier *in silico* les potentielles séquences promotrices reconnues par le facteur σ^{70} de l'ARN polymerase (σ^{80} pour *H.pylori*). Elles sont le fruit d'une collaboration avec Anne Vanet, de l'Institut de Biologie Physico-Chimique de Paris VII. On trouvera de plus amples détails sur ces deux premières parties dans [Vanet et al., 1999] et [Vanet et al., 2000]. Certaines des expériences présentées ici sont issues de ces publications, d'autres ont été réalisées ultérieurement.

Une dernière section présente les autres expérimentations que nous avons réalisées sur des données biologiques.

7.1 Expériences sur des données « propres »

Les expériences décrites dans cette partie ont été réalisées sur des jeux de données propres, dans lesquels on sait que se trouvent des séquences promotrices connues, avec une bonne couverture. Ces expériences ont pour but de tester l'efficacité de SMILE à inférer des modèles structurés d'un jeu de données d'origine biologique, et de vérifier sa capacité à modéliser des sites de fixation dont on connaît assez bien les caractéristiques. Les résultats qu'on obtient sont comparés avec ceux produits par MEME, qui était (et reste à ce jour) un des outils les plus connus et utilisés pour inférer des signaux de tous types.

7.1.1 Conditions des expériences

7.1.1.1 Jeux de données

Nous utilisons deux jeux de données issus des génomes de *Escherichia coli* et *Bacillus subtilis*, baptisés respectivement E_c et E_s. Ils proviennent de [Ozoline et al., 1998] et [Helmann, 1995]. Chacune des séquences de ces jeux est issue d'une zone se trouvant en amont du début de la transcription d'un gène, déterminé expérimentalement. Elles contiennent une séquence promotrice du facteur σ^{70}. Les séquences sont alignées sur le début de la transcription (ce que ni MEME ni SMILE ne prennent en compte). Les caractéristiques de ces deux jeux de séquences sont données par le tableau suivant :

	E_c (*E.coli*)	E_s (*B.subtilis*)
Nombre de séquences	441	131
Nombre de nucléotides	35 115	13 099
Longueur moyenne	79	99
%A	28.3	35.3
%C	20.6	14.2
%G	20.6	17.6
%T	30.6	32.9

Ces jeux ont des tailles relativement réduites par rapport à ceux que nous avons utilisés dans le chapitre précédent.

7.1.1.2 Outils d'inférence

Nos expériences consistent ici à inférer le site de fixation reconnu par le facteur σ^{70} de l'ARN polymérase, connu dans la littérature ([Record et al., 1996]) sous la forme `TTGACAx[15-19]TATAAT`, des jeux de séquences E_c et E_s à l'aide de MEME [1] et SMILE.

Le fonctionnement de MEME nécessite seulement de préciser :
- la couverture supposée du motif à inférer (à choisir entre les options « zéro ou une occurrence par séquence », « une occurrence par séquence », ou « un nombre quelconque d'occurrences par séquence ») ;
- une estimation de la longueur du motif (à préciser exactement ou à choisir entre « court » ou « long ») ;
- le nombre de motifs à produire.

Concernant le dernier point, rappelons que les algorithmes de type EM obtiennent un unique motif au terme de chaque cycle de convergence. Pour obtenir un second motif, il leur faut disqualifier du jeu de séquences les occurrences du premier motif obtenu et effectuer un nouveau cycle complet. Le choix d'un grand nombre de motifs à produire peut par conséquent entraîner un accroissement conséquent du temps de calcul.

[1] une version en ligne de MEME (nos tests utilisaient la version 2.2) est fournie par leurs auteurs à l'adresse `http ://www/sdsc/edu/MEME/meme/website/`. Elle fonctionne sur un super-calculateur de type Cray T3E.

MEME se fie à des critères de signification (l'entropie relative d'une matrice de type *log-likelihood*) pour parcourir l'espace des solutions. Les critères structurels à fournir sont ici très faibles, voire nuls quand on ne précise ni la longueur exacte des motifs à inférer, ni leur couverture. Dans ce cas, ces critères doivent être estimés par l'algorithme.

Pour chaque motif inféré, MEME produit une matrice simplifiée, un graphique indiquant les positions les plus significatives de cette matrice et les symboles correspondant à ces positions. Puis suit une liste des occurrences retenues pour composer cette matrice (que nous ne présentons pas). La signification d'un motif est estimée par la quantité d'information (en bits, à gauche du graphique) contenue par sa matrice, sa longueur et le nombre d'occurrences concernées (champs `width` et `sites`).

SMILE a été utilisé pour inférer d'abord des modèles simples, puis des modèles structurés composés de deux boîtes séparées par un intervalle variable. L'extraction de modèles simples se fait sur des critères variés, par tâtonnement. Nous ne présentons que celles des extractions qui se sont révélées pertinentes. Les extractions de modèles structurés débutent avec des caractéristiques déduites de celles des modèles simples obtenus auparavant.

Les modèles obtenus après extraction sont évalués par simulation préservant la composition en di-nucléotides. L'augmentation de la taille des mots conservés lors des *shufflings* de la simulation donne de moins bons résultats (les modèles qui nous intéressent sont moins en évidence), même pour des tri-nucléotides. Cela s'explique par la faible longueur des séquences utilisées : en augmentant la taille des k-mers, les *shufflings* ont tendance à trop souvent produire des séquences identiques aux séquences originales. Les *shufflings* conservant la composition en nucléotides donnent également des résultats moins pertinents.

Comme lors des tests du chapitre précédent, nous utilisons un Bi-Pentium III cadencé a 1Ghz, avec 512Mo de RAM. Les temps de calcul sont par conséquent difficilement comparables avec ceux de MEME.

7.1.2 Résultats

7.1.2.1 MEME

Les critères d'inférence fournis à MEME pour l'étude des jeux E_c et E_s sont les suivants :
- motifs « courts » (le choix de l'option « longs » produit des résultats équivalents) ;
- couverture de « zéro ou une occurrence par séquence » ;
- inférence de trois motifs au plus (les motifs supplémentaires produits par MEME dans ces expériences ne sont pas significatifs).

Les deux premiers motifs obtenus pour les jeux E_c et E_s sont présentés figures 7.1 et 7.2 respectivement. Les troisièmes motifs produits pour les deux jeux n'étaient pas significatifs.

```
*****************************************************************
MOTIF  1        width =  46 sites = 185.2
*****************************************************************
Simplified     A  44434433211144433222221333443112:9266:22221212
motif letter-  C  2222122322:1242222232333121223221:212:24355454
probability    G  21211222411521212232333232112 3441:221:42333322
matrix         T  2323323326742224333322233333228:411a32111222

         bits 1.7                                      *
              1.5                                      *
Information   1.3                                  *   *
content       1.1                                  *   *
(10.0 bits)   0.9                                 **   *
              0.7           *                     **   *
              0.4           *                     **   *
              0.2          *** *                  ** ***   ** *
              0.0 ----------------------------------------------
Multilevel        AAATAAAAGTTGACATTTTTTGGAGTAAATGGTATAATGCGCCCCC
consensus         CTTATTTCT  TGACAACGCGCCCAATTTGTT  A C T CGGGGA
sequence             C  CTA   C   CACGAATGTC CGCC       A A    T
                        GGC         A C  T    C

*****************************************************************
MOTIF  2        width =  43 sites = 11.0
*****************************************************************
Simplified     A  1541::7::4417753:::48:1:884:2::236:4788::53
motif letter-  C  24:31121344423:41:35:1:11:4::::::2:11::817
probability    G  ::4::3:31:11::5:::3::1361:1:59:753::::1::3:
matrix         T  7:2695155114:::399412763:2:93:9:31761::911:

         bits 1.7
              1.5                              *
Information   1.3                  *         * **        *
content       1.1     *           **         * **        **
(32.9 bits)   0.9     *       *   **  *    * * ***  *  **** *
              0.7  *  *       *** **  ** *** * ***  ******* *
              0.4 ** *******  *** ** ******* * *** ******** *
              0.2 ***************************************** *
              0.0 -------------------------------------------
Multilevel        TAATTTATTCACAAACTTTCATTGAAATGGTGGATTAAATCAC
consensus         CCGC G GCACTCCGA  GA  GT  C T  ATGCA     GA
sequence                         T  C             A
```

Fig. 7.1 – MEME : résultats obtenus sur le jeu E_c.

Le premier motif correspond bien au consensus `TTGACAx[15-19]TATAAT` pour la famille de promoteurs σ^{70} de *E.coli*.

```
************************************************
MOTIF  1        width =  30 sites = 121.0
************************************************
Simplified     A  :1:5353344333345542312319387:6
motif letter-  C  ::12511222222211221211111:1:1::
probability    G  117:1111212122212222253::111:2
matrix         T  982224443434333322426238151192

         bits 1.5
Information   1.3 *                           *
content       1.1 *                       *   *
(11.6 bits)   0.9 **                     **   *
              0.7 ***                    ** * *
              0.4 ***                    ** ***
              0.2 ******              ** *******
              0.0 ------------------------------
Multilevel        TTGACATTATTTTAAAAATATGATATAATA
consensus           TTATAATAAAATTT TGT  G  A   G
sequence             C        CC    AG  T


************************************************
MOTIF  2        width =  30 sites = 21.8
************************************************
Simplified     A  22:53552:261266956111:44789882
motif letter-  C  1431134313121:1:2:32:41521:::2
probability    G  4111211:331222::22:11:111:1::4
matrix         T  335341:4632542211276863::::221

         bits 1.5
Information   1.3                           *
content       1.1                *          *
(16.1 bits)   0.9                *         ****
              0.7                *  * ** ******
              0.4   *   * *    *** ***** ******
              0.2   ** **** ** ********* *******
              0.0 ------------------------------
Multilevel        GCTATAATTCATTAAAAATTTTACAAAAAG
consensus         TTCTACCCGT  G T C CC CTAC    C
sequence          A   G  A G                   A
                           A
```

FIG. 7.2 – MEME : résultats obtenus sur le jeu E_s.
Le premier motif correspond également au consensus `TTGACAx[15-19]TATAAT` pour la famille de promoteurs σ^{70} de *B.subtilis*.

Les temps nécessaires à MEME pour produire ces résultats sont de 21mn 15sec pour E_c, et de 3mn 4sec pour E_s. Les premiers motifs reportés pour les deux jeux de séquences correspondent aux séquences promotrices déterminées expérimentalement pour les deux organismes. Dans les deux cas, le consensus `TTGACAx[15-19]TATAAT` est assez visible.

Chez *E.coli*, ce premier motif recueille 185 occurrences (42% des séquences sont concernées) dont la plupart sont correctes. Les positions de la première partie de ce premier motif (que nous appellerons boîte mais qui n'en est pas vraiment une pour MEME) sont relativement mal conservées. Par contre, les trois positions les plus fortes de la seconde boîte sont celles qui sont connues comme étant les mieux conservées. La distance qui sépare les deux boîtes détectées est de 18 bases.

Le second motif obtenu pour *E.coli* couvre moins de 7% des séquences et ses occurrences ne sont pas alignées par rapport aux débuts de transcription. Il est donc peu probable qu'il représente un site pertinent, et ce malgré son fort contenu informationnel (32.9 bits). Rappelons que l'entropie relative utilisée ici ne prend pas en compte la couverture des motifs, et peut donc donner, comme c'est le cas dans cet exemple, un meilleur score à un motif très conservé apparaissant dans peu de séquences qu'à un motif mal conservé apparaissant dans un grand nombre de séquences.

Chez *B.subtilis*, le meilleur motif recueille 121 occurrences (92% des séquences), la plupart correctes. Ici, seules deux des trois positions les mieux conservées de la TATA-box sont reportées. La première position de la TATA-box obtient un score faible, mais les deux positions qui la précèdent (souvent considérées comme faisant partie d'une TATA-box étendue) sont relativement fortes. La distance séparant les deux boîtes est de 15 nucléotides.

Le second motif obtenu possède des occurrences dans 16% des séquences, et ne semble pas être correct malgré le faible motif de la forme TATAAT décrit : 13 des 20 occurrences seulement correspondent effectivement à une TATA-box.

7.1.2.2 SMILE

E.coli

SMILE a d'abord été utilisé pour inférer des modèles simples d'au moins 6 bases avec une substitution. Après quelques tâtonnements rapides, on obtient, avec un quorum de 25%, un résultat dont les dix meilleurs modèles (parmi les 2452 produits) sont présentés figure 7.3. L'extraction est faite en moins d'une demi-seconde, et l'évaluation en trente secondes. La TATA-box apparaît dans plus de 60% des séquences et se place en quatrième position. Des extensions de la TATA-box (`TATAATG, ATATAAT, TTATAAT, TATAATT` apparaîssent également, mais avec des scores nettement plus faibles.

La première boîte du consensus connu pour la famille σ^{70} de *E.coli* n'apparaît, elle, qu'en 80$^{\text{ème}}$ position. Elle se trouve en revanche aux troisième et septième positions sous des formes décalées : `TGACAC` et `CTGACA`.

Model	%right	#right	%shfl.	#shfl.	Var.	Chi2	Z-score
GATAAAA	30.39%	134	18.40%	81.13	7.65	17.18	6.91
GATAAA	56.69%	250	42.90%	189.19	9.07	16.77	6.70
TGACAC	39.46%	174	26.71%	117.77	8.82	16.19	6.38
TATAAT	61.68%	272	49.61%	218.80	8.40	13.00	6.33
CCTGAA	38.10%	168	27.30%	120.41	7.79	11.67	6.11
ATACTG	42.40%	187	31.52%	138.99	8.15	11.22	5.89
CTGACA	36.05%	159	25.66%	113.15	7.88	11.17	5.82
GAAAAAA	27.66%	122	19.22%	84.77	6.41	8.76	5.81
ACACTG	38.10%	168	26.57%	117.17	8.77	13.39	5.80
ATAAGG	40.59%	179	29.13%	128.46	8.78	12.75	5.76
...							

FIG. 7.3 – SMILE : inférence d'une boîte sur E_c.
Quorum 25%, longueur minimum 6, et une substitution. Seuls les dix meilleurs modèles produits au terme de la simulation sont présentés. La TATA-box est en quatrième position, et une variante de la première boîte du consensus `TTGACAx[15-19]TATAAT` en septième position.

Nous débutons l'inférence de modèles structurés composés de deux boîtes par une extraction des modèles de longueur 6 pour chaque boîte, avec deux substitutions chacune pour un total de trois substitutions au plus, et un quorum de 25%. La distance entre les deux boîtes est balayée grâce aux deltas : nous choisissons d'explorer tous les intervalles de rayon 1 dans l'intervalle $[8; 24]$. Cette inférence doit permettre de déterminer l'intervalle de distance le plus pertinent entre les deux boîtes.

La figure 7.4 présente une courbe indiquant les meilleurs modèles obtenus et leur z-score, en fonction de l'intervalle de distance qui sépare les boîtes. L'extraction et l'évaluation pour l'ensemble de ces intervalles de saut sont réalisées en 2h 30mn, dont une grande partie pour l'extraction. Très peu de modèles sont produits à l'issue des extractions, ce qui rend le travail d'évaluation rapide. Ce faible nombre de modèles obtenu lors de l'extraction (entre zéro et dix-neuf selon l'intervalle) met en avant le fait que peu de modèles structurés remplissent les conditions requises, et que la séquence promotrice a déjà, par sa structure, un caractère exceptionnel.

Les résultats de l'évaluation mettent clairement en avant le consensus attendu, pour les intervalles $[15; 17]$ et $[16; 18]$, ce qui semble indiquer que l'intervalle $[15; 18]$ lui est spécifique. Parmi les meilleurs modèles obtenus avec un score elevé, celui de l'intervalle $[17; 19]$ est également très proche de ce consensus.

Supposant que l'intervalle de validité du consensus est donc $[15; 19]$, nous relançons une extraction pour cet intervalle uniquement, avec un quorum de 30% car l'intervalle de saut plus large doit permettre d'obtenir d'avantage de modèles. Après une première approche

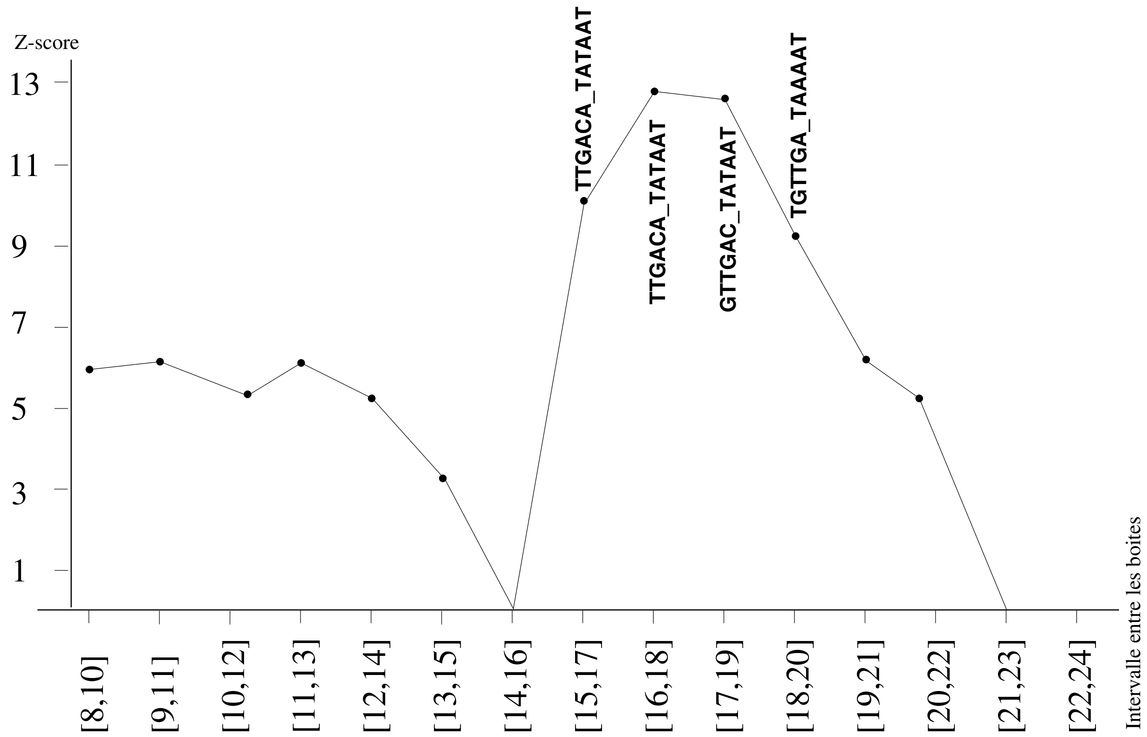

FIG. 7.4 – SMILE : inférence de deux boîtes sur E_c, par balayage de l'intervalle $[8;24]$ avec un $\delta = 1$.

Quorum 25%, longueur 6 pour chaque boîte, deux substitutions par boîte et trois substitutions globales. En fonction de l'intervalle de saut entre les deux boîtes, le z-score du meilleur modèle obtenu est représenté. Le modèle correspondant est indiqué pour les z-scores les plus élevés.

large qui nous a permis d'isoler les intervalles significatifs, cette phase correspond à une recherche plus fine du consensus. Dans l'éventualité de la présence d'un consensus étendu, nous réalisons le test qui suit en autorisant qu'une des deux boîtes ait éventuellement une longueur 7.

L'extraction et l'évaluation sont réalisées en 24mn, et on obtient 19 164 modèles dont les résultats d'évaluation sont présentés à la figure 7.5.

Le résultat de cette inférence est catégorique, le consensus `TTGACAx[15-19]TATAAT` obtient le meilleur score (de loin pour les χ^2) et une couverture de 36% des séquences. Aucune boîte d'aucun modèle n'a une longueur supérieure à 6 bases.

Ces résultats semblent indiquer, comme ceux de MEME, que la troisième position de la TATA-box peut contenir un A ou un T. Le G qui précède la TATA-box semble également pertinent, mais aucune première boîte GTTGACA n'apparaît.

Model	%right	#right	%shfl.	#shfl.	Var.	Chi2	Z-score
TTGACA_TATAAT	36.96%	163	17.39%	76.67	7.69	42.70	11.22
GTTGAC_TAAAAT	30.61%	135	14.83%	65.40	6.30	31.28	11.06
GTTGAC_TATAAT	31.07%	137	13.38%	59.01	7.28	39.90	10.71
TTTCAC_TATAAT	34.24%	151	17.86%	78.75	7.14	30.72	10.12
TTTGAC_TATAAT	33.79%	149	18.27%	80.55	7.05	27.59	9.71
GTTGAT_TATAAT	32.65%	144	16.55%	73.00	7.45	30.81	9.54
GTTTTC_TATAAT	32.65%	144	17.53%	77.29	7.58	26.85	8.80
TTGATA_TATAAT	33.79%	149	19.62%	86.52	7.40	22.61	8.45
TTGACA_TAAAAT	34.24%	151	18.78%	82.84	8.13	27.04	8.38
TGACAT_TATAAT	30.39%	134	16.62%	73.31	7.28	23.23	8.34
...							

FIG. 7.5 – SMILE : inférence de deux boîtes sur E_c, avec l'intervalle $[15; 19]$.
Quorum 30%, longueur 6 pour chaque boîte, trois substitutions globales et deux par boîte. Les dix meilleurs modèles sont présentés.

B.subtilis
Dans les mêmes conditions que précédemment (quorum de 25% et longueur d'au moins 6 bases, avec une substitution), les meilleurs modèles simples obtenus (parmi les 3 117 modèles extraits et évalués en moins de 20 sec.) pour le jeu E_s sont donnés figure 7.6.

Bon nombre des meilleurs modèles produits sont des décalages et/ou des extensions d'une des deux boîtes du consensus connu `TTGACAx[15-19]TATAAT`. Les deux boîtes exactes de ce consensus n'apparaissent respectivement qu'aux 241$^{\text{ème}}$ et 102$^{\text{ème}}$ positions avec des significations statistiques faibles. On en conclut dès lors que, dans ce jeu de séquences, le consensus de la famille de promoteurs σ_{70} (en en particulier la TATA-box) pourrait être plus étendu.

Nous effectuons ensuite une inférence des modèles structurés dont chacune des deux boîtes a une longueur 6, avec 2 substitutions globales et 2 par boîte, par balayage sur l'intervalle $[8; 24]$ avec un delta de 1, et un quorum de 30%. Le résultat est obtenu en 7mn 5sec, et est présenté figure 7.7.

Comme dans le cas de *E.coli*, on obtient clairement un large pic dans la signification statistique, pour les modèles ayant un saut compris entre 15 et 19 bases. Les meilleurs modèles recueillis correspondent au consensus connu. Nous retenons donc cette longueur de *spacer* entre les boîtes.

Afin d'affiner le consensus et de viser une couverture plus large des séquences, nous inférons les modèles de mêmes caractéristiques sur l'intervalle $[15; 19]$ avec un quorum de 70%. En 4mn 46sec. on obtient les résultats de la figure 7.8. On trouve le consensus attendu avec

Model	%right	#right	%shfl.	#shfl.	Var.	Chi2	Z-score
GGTATAAT	28.24%	37	6.86%	8.99	3.00	20.69	9.33
GTATAATA	35.88%	47	10.67%	13.98	3.58	23.30	9.23
TATAATA	71.76%	94	39.46%	51.69	4.68	27.68	9.04
TGTTATAA	35.88%	47	14.05%	18.41	3.71	16.65	7.71
GTTATAAT	32.06%	42	12.40%	16.25	3.37	14.64	7.64
TGTTGAC	33.59%	44	13.34%	17.48	3.48	14.95	7.61
TATAATAAT	27.48%	36	7.92%	10.38	3.39	17.20	7.55
GTTGACA	34.35%	45	14.40%	18.86	3.54	14.15	7.39
GTATAAT	56.49%	74	28.69%	37.58	4.96	20.71	7.34
TATAATAT	38.93%	51	17.27%	22.62	4.07	15.22	6.97
...							

FIG. 7.6 – SMILE : inférence d'une boîte sur E_s.
Quorum 25%, longueur minimum 6, et une substitution. Ces dix premières positions (et bon nombre des suivantes) sont exclusivement composées d'une des deux boîtes du consensus connu, étendues à droite ou à gauche.

plusieurs positions variables sur la première boîte et une seconde boîte qui semble pouvoir être étendue, d'après ces résultats et ceux obtenus lors de l'inférence de modèles simples. Pour vérifier cette hypothèse, nous réalisons donc une inférence de modèles dont une des deux boîtes serait étendue à 7 bases (figure 7.9, résultats obtenus en 9mn 23sec). Cette dernière expérience fait état de la présence dans près de 60% des séquences d'un consensus étendu de la forme `TTGACAx[15 ;19]TATAATA`, obtenant un score élevé. Elle confirme également que la première boîte du consensus subit plus de variations que la TATA-box.

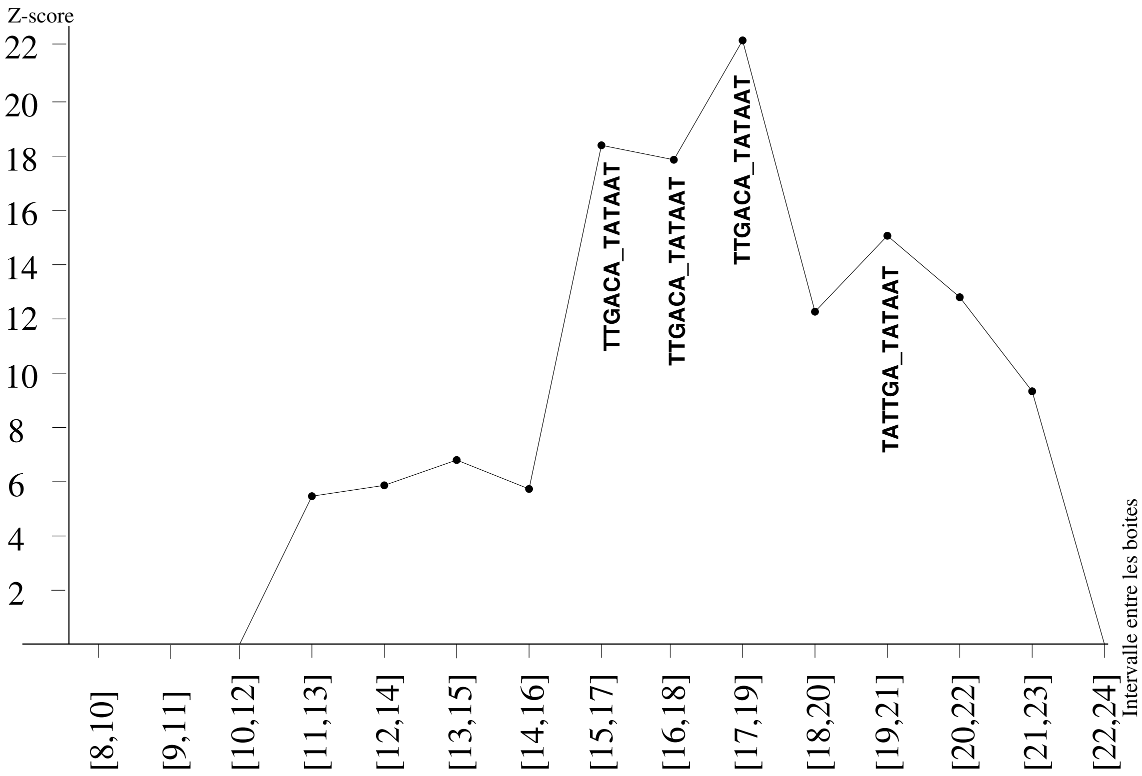

FIG. 7.7 – SMILE : inférence de deux boîtes sur E_s, par balayage de l'intervalle $[8; 24]$ avec $\delta = 1$.

Quorum 30%, longueur 6 pour chaque boîte, deux substitutions par boîte et deux substitutions globales.

Model	%right	#right	%shfl.	#shfl.	Var.	Chi2	Z-score
TTGTCA_TATAAT	74.05%	97	35.54%	46.56	5.14	39.20	9.81
TTGACA_TATAAT	73.28%	96	37.81%	49.53	4.93	33.38	9.42
TTTACA_TATAAT	80.15%	105	45.05%	59.02	4.91	34.47	9.37
TTGACA_ATAATA	71.76%	94	36.78%	48.18	4.94	32.29	9.27
TTTGAC_TATAAT	70.99%	93	36.94%	48.39	5.07	30.57	8.80
TTTGCA_TATAAT	70.23%	92	38.46%	50.38	4.75	26.65	8.75
TTGATA_TACAAT	70.99%	93	39.08%	51.20	4.93	26.95	8.47
TTTACA_TATTAT	71.76%	94	42.79%	56.06	4.62	22.45	8.21
TTGATA_TAATAT	72.52%	95	47.54%	62.28	4.08	17.03	8.01
TTTACA_ATATTA	74.05%	97	46.07%	60.35	4.59	21.37	7.98
...							

FIG. 7.8 – SMILE : inférence de deux boîtes sur E_s, avec l'intervalle $[15; 19]$. Quorum 70%, longueur 6 pour chaque boîte, deux substitutions par boîte et trois substitutions globales.

Model	%right	#right	%shfl.	#shfl.	Var.	Chi2	Z-score
TTGCCA_TTATAAT	50.38%	66	11.81%	15.47	2.95	45.48	17.14
TTGACA_TATAATA	58.02%	76	17.35%	22.73	3.75	46.12	14.20
TTGACA_GTATAAT	51.15%	67	12.21%	16.00	3.60	45.87	14.18
TTGACT_TATAATA	55.73%	73	15.84%	20.75	4.05	45.35	12.91
TTGACAT_ATAATA	53.44%	70	16.49%	21.60	3.89	39.32	12.43
TTCACA_TATAATA	51.15%	67	15.26%	19.99	3.87	38.03	12.14
ATTGTC_TATAATA	50.38%	66	15.92%	20.85	3.73	35.11	12.09
TTGACAA_ATAATA	53.44%	70	19.05%	24.95	3.75	33.52	12.02
ATTGAC_TATAATA	51.15%	67	16.47%	21.58	3.80	35.19	11.94
TTTACA_TATAATA	61.07%	80	22.22%	29.11	4.27	40.67	11.91
...							

FIG. 7.9 – SMILE : inférence de deux boîtes étendues sur E_s, avec l'intervalle $[15; 19]$. Quorum 50%, longueur totale 13, deux substitutions par boîte et trois substitutions globales.

7.1.3 Discussion

Dans ces conditions d'expérimentation particulières, MEME et SMILE font montre de disparités dans leurs performances et leurs résultats. MEME est bien plus simple à utiliser et offre des temps de réponse assez rapides. Les résultats obtenus sont corrects, bien que des ambiguïtés sur certaines positions semblent provenir de son incapacité à percevoir une distance variable entre les deux motifs significatifs. Grâce à la souplesse permise par la représentation matricielle, la couverture des sites détectés est très bonne. La sensibilité de la représentation matricielle apporte cependant une perte de spécificité.

SMILE fait preuve d'une complexité d'utilisation plus importante, nécessitant certaines déductions de l'utilisateur afin de produire des requêtes pertinentes. Les temps de calcul sont plus élevés, quoique corrects en tenant compte du fait que l'algorithme est exact et que nos tests ont été effectués sur un micro-ordinateur. Les résultats coïncident bien avec le consensus attendu, et mettent nettement en avant les sites à détecter. Compte tenu de la rigidité reprochée à la représentation par motifs consensus, par rapport aux matrices, les couvertures obtenues pour les sites à détecter, parfois très proches de celles obtenues par MEME, sont très encourageantes. Elles confirment que l'ajout de caractéristiques telles que la gestion des *spacers* dans la représentation par motifs lui apporte un supplément de sensibilité. Si cette représentation s'avère être un peu moins sensible que les matrices utilisées par MEME, elle semble néanmoins plus spécifique que celles-ci.

Notons enfin que les deux boîtes clairement mises en avant dans les deux cas sont beaucoup moins visibles par SMILE quand on les considère indépendamment (en particulier pour la première boîte). L'utilisation de modèles structurés est donc, dans notre cas, indispensable et pertinente pour leur détection.

7.2 Expériences sur des données brutes

Les expériences précédentes ont permis de vérifier les capacités d'inférence des outils testés, dans des conditions considérées comme bonnes : le site à inférer est déjà connu et les séquences sont soigneusement selectionnées afin de réduire le bruit. Par conséquent, la couverture du site est importante et la quantité de données réduite.

Ces conditions ne correspondent cependant pas à celles auxquelles sont en général confrontés les bio-informaticiens souhaitant inférer *in silico* un site de fixation. Il est rare d'obtenir de tels jeux de données (ceux utilisés proviennent d'organismes particulièrement étudiés dont certains mécanismes de régulation sont bien connus). Un de nos objectifs d'origine était justement de concevoir un outil capable de travailler dans des conditions plus difficiles, afin d'observer si l'on peut inférer des sites dans des jeux de données brutes, directement issues des banques.

Cette section décrit de nouvelles expériences dans des conditions plus sévères. Les jeux de séquences utilisés sont directement issus des génomes complets de *E.coli*, *B.subtilis* et *H.pylori*. Les annotations de ces génomes nous permettent simplement de restreindre le jeu de données aux régions susceptibles de contenir des signaux jouant un rôle dans l'initiation de la transcription. La question que posent ces expérimentations est la suivante : « peut-on détecter certains sites actifs dans un génome sans aucune autre information que son annotation ? ».

Si, comme nous l'avons vu précédemment, les promoteurs associés au facteur σ^{70} chez *E.coli* et *B.subtilis* sont assez bien connus, ces expériences sur leurs génomes « entiers » nous placent dans des conditions équivalentes à celles rencontrées en terrain inconnu. Nos expérimentations portent également sur l'inférence de promoteurs associés au facteur σ^{80} chez *H.pylori*, qui, eux, ne sont pas connus. Cet organisme vivant dans une niche biologique très précise (l'epithélium gastrique des primates) serait susceptible de posséder des promoteurs différant quelque peu de ceux des autres bactéries.

7.2.1 Conditions des expériences

7.2.1.1 Jeux de données

Les expériences qui suivent ont été réalisées sur des jeux de données issus des génomes complets de *Escherichia coli* (obtenus à l'adresse http ://mol.genes.nig.ac.jp/ecoli/), *Bacillus subtilis* (ftp ://ncbi.nlm.nih.gov/genbank/genomes/bacteria/Bsub/) et *Helicobacter pylori* (ftp ://ftp.tigr.org/pub/data/h_pylori), respectivement notés G_c, G_s et G_p. Grâce aux annotations fournies, nous avons retenu les zones non-codantes se trouvant entre deux gènes divergents [2], et sélectionné parmi ces zones les amonts des deux gènes considérés. Les séquences obtenues sont donc non-codantes sur les deux brins de l'ADN, augmentant ainsi les chances d'y trouver un promoteur. Les séquences de moins de 40 bases ont été retirées du jeu, en supposant qu'il était peu probable qu'elles contiennent des promoteurs. Les séquences restantes ont été tronquées au-delà des 330 nucléotides qui précèdent le début de la traduction annoté, afin d'éliminer les zones où l'on ne pense plus trouver les promoteurs qui nous intéressent. Enfin, les 15 bases qui précèdent le début de la traduction ont été retirées afin d'éliminer le motif de Shine-Dalgarno [3] qui s'y trouverait.

Le tableau qui suit indique les caractéristiques de ces trois jeux :

	G_c (*E.coli*)	G_s (*B.subtilis*)	G_p (*H.pylori*)
Nombre de séquences	1 148	1 062	308
Nombre de nucléotides	226 928	196 736	52 100
Longueur moyenne	197	185	169
%A	29.9	33.1	35.7
%C	20.1	16.8	14.2
%G	19.9	17.1	14.5
%T	30.1	33.0	35.6

Ces jeux n'offrent évidemment pas les mêmes garanties de couverture par des promoteurs que les jeux E_c et E_s déterminés expérimentalement. Notons que les deux premiers jeux

[2] qui sont exprimés sur des brins différents de l'ADN, et donc dans des directions différentes.

[3] fréquemment rencontré dans les régions précédant le début de la traduction (voir la figure 1.3 page 27).

sont de tailles conséquentes par rapport à ceux manipulés précédemment.

7.2.1.2 Outils d'inférence

À l'époque où ces expériences ont été réalisées, nous avions utilisé MEME pour comparer les résultats que nous obtenions. Pour les deux plus gros jeux de données, nous nous étions heurtés à une limitation imposée sur la version en ligne de MEME : les jeux devaient contenir moins de 100 000 symboles. Faute de mieux, nous avions construits deux jeux Gr_c et Gr_s à partir des jeux originaux, en sélectionnant aléatoirement une partie de leurs séquences. Nous retranscrivons ici les résultats obtenus à l'époque [4].

Nous testons également un outil plus récent, Motifs Sampler, qui utilise un algorithme d'optimisation du type *Gibbs sampling* dont le principe est décrit dans [Thijs et al., 2001b]. Si celui-ci ne fixe pas de limitation dans la taille du jeu utilisé, certaines expériences sur des jeux trop gros n'ont pas pu aboutir.

Des dires mêmes de l'un de ses auteurs, Gert Thijs, Motifs Sampler ne peut probablement pas détecter des motifs ayant une trop faible couverture dans un jeu de séquences trop large. Le fait que les motifs recherchés soient structurés rend cette détection encore plus délicate : même si le *spacer* entre deux motifs était de longueur fixe, ce que nous appelons un modèle structuré est perçu par Motifs Sampler comme un motif simple contenant un grand nombre de positions faibles, et donc très difficile à détecter.

Contrairement à celles utilisant MEME, les expériences sur Motifs Sampler ne se veulent en aucun cas exhaustives ; notre but est uniquement de montrer comment il fonctionne et le type de résultats qu'il produit en utilisant des contextes d'ordre plus elevé que la plupart des outils d'optimisation.

SMILE est utilisé dans les mêmes conditions que précédemment. La longueur des k-mers conservés lors des *shufflings* de la simulation donne lieu à quelques expérimentations, présentées dans l'annexe A, page 203, qui visent à observer leur influence sur les résultats de l'évaluation.

7.2.2 Résultats

7.2.2.1 MEME

Les critères choisis sont les mêmes qu'auparavant : motifs « courts », « zéro ou un motif par séquence » et trois motifs à produire. La modification du paramètre de longueur ne donne pas de résultats plus significatifs.

E.coli
L'utilisation de MEME avec le jeu réduit Gr_c ne produit pas de consensus statistiquement significatif ressemblant à la séquence promotrice attendue (les deux meilleurs motifs

[4] la dernière version en ligne (3.0) possède une limitation plus faible (60 000 bases).

produits sont présentés figure 7.10). Le fait que l'inférence ait pu aboutir prouve que le problème ne provient pas tant de la taille du jeu de données que de la faible couverture du motif à détecter. Le meilleur motif produit correspond cependant bien au site de fixation connu (`TGTGAx[6]TCACA`) pour le facteur de transcription CRP de *E.coli* ([Combrugghe et al., 1984]). Le second motif, malgré un contenu informationnel du même ordre que le précédent et une large couverture, ne paraît pas pertinent.

```
*********************************************************
MOTIF  1        width =  30 sites = 111.4
*********************************************************

         bits 1.5
Information   1.3              *          *
content       1.1            * *        * *
(12.3 bits)   0.9            * *        * *
              0.7            ***       ****
              0.4         *  ****      *****
              0.2       *** *****      ***** **
              0.0 ------------------------------
Multilevel        AATTTTAAATTGTGATCTAAATCACATATT
consensus         CGAAGATTTA  C   AGTGT  G  ATAA
sequence          G G             TAGT         G

*********************************************************
MOTIF  2        width =  39 sites = 128.7
*********************************************************

         bits 1.1
Information   0.9                                  *
content       0.7  *                    *          *
(12.0 bits)   0.4  ** * * *           * *      **  * *
              0.2  **** ****    *  *  * *    ****  * ** *
              0.0 ---------------------------------------
Multilevel        TAATTAATATACACAATTTTTTTTTTATTTTCATGATTT
consensus         AC AATTATCTAGT TAAAACAAGAATAAA AT TCAAA
sequence          C    C    CGTA  C GG G A  C    T  C
                              C
```

FIG. 7.10 – Résultats obtenus par MEME sur le jeu Gr_c.
Le premier motif correspond vraisemblablement au site de fixation CRP.

B.subtilis
L'inférence réalisée sur le jeu réduit Gr_s ne donne aucun résultat significatif (figure 7.11). Les motifs produits ont un faible contenu informationnel et/ou peu de positions conservées. Le troisième motif obtenu présentait une suite de T peu conservés.

```
*****************************************
MOTIF  1        width =  12 sites = 308.7
*****************************************

         bits 1.1      **
Information   0.9      **
content       0.7      **
(5.6 bits)    0.4     ****** *
              0.2    *********
              0.0 ------------
Multilevel        AAAAAAAGGAGG
consensus         TGG    ACGAA
sequence          CT     T   T

*****************************************
MOTIF  2        width =  22 sites = 54.4
*****************************************

         bits 0.7  **    *             *
Information   0.4 ***  * **       *    *
content       0.2 ****** *****  * **  **
(12.3 bits)   0.0 ----------------------
Multilevel        GGCAGCAGCCCGTGCAGAGCGA
consensus          C T  C AAA GAATACCGAG
sequence             G  T     A   CTAAC
```

FIG. 7.11 – Résultats obtenus par MEME sur le jeu Gr_s.

H.pylori
Avec le jeu complet G_p, MEME obtient les résultats présentés figure 7.12. Le premier motif recueille un score élevé malgré un faible nombre d'occurrences, mais ne semble pas pertinent. Le second motif, présente un motif proche d'une TATA-box, ayant un grand nombre d'occurrences.

```
*********************************************************
MOTIF  1        width =  28 sites = 22.4
*********************************************************

         bits 1.7              **
              1.5              **     *  *
Information   1.3              **  *  *  * *
content       1.1              **  *  *  * *
(19.1 bits)   0.9              *** ****  ***
              0.7 *            *** ***** ***
              0.4 **  *      **************
              0.2 **  *   *  ****************
              0.0 ----------------------------
Multilevel        GGTGGGGTGTCGGGGGTTCGAACCCCCG
consensus         T CTTTAGTGGA   A  T   T   TG
sequence               AC  AA
                       C
*********************************************************
MOTIF  2        width =  39 sites = 274.0
*********************************************************

         bits 1.1                        *
Information   0.9                     **  *
content       0.7                     ** **
(12.3 bits)   0.4    *           *    ** ** *      ***
              0.2   *****       ***   ** ** *      ******
              0.0 ---------------------------------------
Multilevel        TAATAAAAAGCAATTTTTTGCTAAAATAATGCTTTTTAA
consensus         ATTATTTTCATTTAAAAACAT  T  A GCATC  AATT
sequence          CC      TC          A       CACA
                           T                   G
```

FIG. 7.12 – Résultats obtenus par MEME sur le jeu *Gp*.
Le second motif pourrait contenir une TATA-box.

7.2.2.2 Motifs Sampler

Motifs Sampler fonctionne avec les mêmes types de paramètres que MEME. Il faut lui fournir la longueur (fixée) des motifs à inférer, ainsi qu'une couverture maximale par séquence (que nous fixons à 1) et un nombre de motifs à produire (fixé à 6). Il faut aussi paramétrer l'ordre du modèle de Markov qui décrit le contexte à partir des séquences fournies (nous avons choisi l'ordre 3, paramétré par défaut). Enfin, le recouvrement maximum autorisé entre les motifs obtenus était fixé à 2.

E.coli
Le jeu G_c s'avère être trop conséquent pour Motifs Sampler.

B.subtilis
Lors de l'extraction de motifs de longueur 6 sur le jeu complet G_s, Motifs Sampler produit les résultats suivants :

Consensus	Score	Entropie relative	*Log-likelihood*	Nb occurrences
ATyATA	1.559	1.279	2868.492	634
TACnAT	1.401	1.276	2128.002	471
TCCwTT	1.360	1.258	2621.676	647
ATTsAy	1.309	1.254	2348.899	517
TGTTTA	1.358	1.249	2258.734	501
yyTAAA	1.411	1.237	2129.806	472

Le premier motif obtenu pourrait être un décalage de la TATA-box, mais le symbole Y (C ou T) ne représente pas le A attendu. Le quatrième motif ATTsAy pourrait, lui, être un décalage de la boîte TTGACA. Les autres motifs ne semblent pas décrire les boîtes du consensus attendu.

Avec une longueur de 8, on obtient :

Consensus	Score	Entropie relative	*Log-likelihood*	Nb occurrences
yTCnTTTT	1.106	1.051	2270.033	638
TrwCAAwA	1.100	0.985	2011.816	487
TArAAAAA	1.141	0.966	2583.828	716
AAwTGGww	1.026	0.950	2149.181	543
TnATTwTA	1.184	0.935	2883.198	745
yryAnrnT	0.921	0.850	2608.206	787

Les troisième et cinquième motifs (qui ont les scores les plus élevés) pourraient décrire la TATA-box. Comme précédemment, les autres motifs ne semblent pas décrire les boîtes du consensus attendu.

Motifs Sampler n'a pas pu conclure l'inférence de motifs de longueur 29, visant à détecter les deux boîtes du consensus. L'expérience d'un de ses auteurs indique que l'inférence de motifs aussi longs dans de si grands jeux de séquences ne donne en général aucun résultat.

H.pylori
Le jeu G_p a été traité par Gert Thijs, qui a réalisé une série d'exécutions visant à mettre en évidence des motifs vers lesquels le processus stochastique convergerait fréquemment. Au lieu d'utiliser les séquences fournies pour construire le contexte comme précédemment, Gert Thijs a modélisé le contexte par une chaîne de Markov d'ordre 3 pré-calculée à partir de toutes les séquences de longueur 1000 se trouvant en amont des gènes de *H.pylori*. Le recouvrement maximum des motifs obtenus était fixé à 1.

La première inférence a été réalisée 50 fois, pour des motifs de longueur 6, produisant trois motifs à chaque fois. Les 150 motifs obtenus ont été classés selon leur *log-likelihood ratio*. Le motif TATAAT occupe les 20 meilleures positions parmi les motifs obtenus, avec près de 255 occurrences relevées à chaque fois. Le reste des résultats est majoritairement composé de motifs TAwAAT, ATTATA et ATTAwTA. La TATA-box semble donc être facilement mise en évidence.

La seconde série d'inférences visait à détecter deux boîtes. Nous avons donc essayé d'inférer un motif de longueur 33 (longueur déduite des résultats obtenus par SMILE, présentés plus loin). L'opération a été répétée 100 fois, avec les mêmes paramètres qu'auparavant, produisant 300 motifs qui ont été triés selon leur entropie relative...

Consensus	Entropie relative	Nb occ.
TTAAAnTACAAACAACAGCATAATGGCAACATG	1.080	3
TTTAAAnTACAAACAACAGCATAATGGCAAAAT	0.948	3
TTGTGwTkTTTwTCTTTmwTATwwCynwAwrGw	0.762	6
AwTnTnwwknnwkTTTTnTkTwTTATnkTkTTT	0.698	11
TAATAnwnTnnnwwAnnArynTwnTTwnmTTyA	0.693	4

...ou leur *log-likelihood ratio*...

Consensus	*Log-likelihood*	Nb occ.
TwnwnnwwwTTwTrnTATAATnnnnnnwwwwww	1946.697	255
TwnwnnwwwTTwTrnTAnAATnnwnnwwwwwww	1938.408	251
TwnwnnwwwTTwTrnTAnAATwnnnnywwwwww	1937.600	259
TwnwnnwwwTTwTrnTAwAATwnnnnnwwwwww	1926.615	255
TwnwnnwwwTTwTrnTAnAATwnnnnnwwwwww	1917.781	255

Les motifs obtenant les meilleures entropies relatives ont de très faibles couvertures. Les deux meilleurs motifs sont très proches et présentent beaucoup de positions conservées, ce qui justifie probablement leur bon score. La TATA-box n'y apparait pas. Le *log-likelihood ratio* qui ne pondère pas les scores de chaque position du motif avec la fréquence du symbole concerné, met en avant des motifs beaucoup plus dégénérés ayant un grand nombre d'occurrences. Les meilleurs motifs obtenus sont identiques et montrent une TATA-box précédée de plusieurs positions conservées.

7.2.2.3 SMILE

Les résultats présentés dans cette section sont directement issus de [Vanet et al., 1999] et [Vanet et al., 2000], excepté la dernière inférence effectuée sur *H.pylori*. À l'époque où nous avons publié ces articles, l'évaluation par les z-scores n'avait pas encore été implantée dans SMILE. C'est pourquoi les résultats issus de ces publications ne présentent que des valeurs de χ^2.

E.coli
On utilise le jeu complet G_c. L'arbre de tous les suffixes généré pèse 24Mo en mémoire. Une première inférence de modèles simples de longueur minimum 6 avec une substitution et un quorum de 40% est réalisée. La simulation consiste en mille *shufflings*. En 18mn 20sec (dont une demi-seconde pour l'extraction !) on obtient 4006 modèles, dont les plus significatifs sont présentés dans le tableau 7.13.

Les résultats obtenus ne font pas apparaître de consensus connu : si la famille 1 pourrait coïncider avec le site connu en -35 (un C remplacerait alors le premier T pour former le consensus `TTGACA`), la TATA-box n'est pas exprimée de façon significative (533ème position). Globalement, les valeurs de χ^2 sont assez faibles.

Espérant que l'association de deux boîtes donne des résultats plus proches de ceux attendus grâce au renforcement des critères structurels, nous réalisons l'inférence de modèles structurés composés de deux boîtes de longueur 6 à 7 pour une longueur totale de 13, avec un quorum de 2% et une substitution globale. Ce faible quorum doit être compensé par la spécificité structurelle du modèle recherché. Le *spacer* est choisi dans la fourchette [4 ;24] avec un $\delta = 1$. La simulation réalise cent *shufflings*. Les résultats de cette inférence (36 039 modèles) sont obtenus en près de 12h (dont 36mn pour l'extraction) et présentés figure 7.14.

Malgré le quorum faible et le grand nombre de modèles obtenu, aucun modèle structuré ne recueille de score élevé dans les intervalles attendus ([15 ;19]). On observe cependant des valeurs significatives dans l'intervale [4 ;7]. Le modèle recueilli, de la forme `TGTGATx[4 ;7]TCACAT`, correspond bien au consensus `TGTGAx[6]TCACA` connu pour le site de fixation du facteur CRP.

Les inférences plus souples que nous avons réalisées, ajoutant substitutions ou jokers, ne donnent pas de résultats plus concluants.

Sur ce jeu de données, SMILE, comme les outils auxquels nous le comparons, ne parvient donc pas à extraire les signaux attendus.

Famille	Modèle	Nb seq	Nb seq shff	χ^2
	CCTGAC	573	424	39
	CTGACG	587	439	38
1	CTGACA	701	557	36
	TCCTGA	671	538	30
	CCCTGA	575	446	28
	GTCAGG	576	412	47
	TGTCAG	702	555	37
2	CATCAG	711	574	32
	CGTCAG	580	443	32
	ATCAGG	689	553	32
	TTTTCTG	553	419	31
	CTCTTTT	464	348	25
3	TTTCTGT	469	357	23
	TTTTCAG	531	416	23
	CTGATTT	498	384	23
	CAGAAAA	539	410	29
	CTGAAAA	525	407	24
4	GAGAAAA	460	359	19
	AGATAAA	512	415	16
	GTGAAAA	509	414	16
	CCGGCA	604	480	26
	CAGGAA	717	595	26
5	CAGGCT	609	490	24
	CAGCAG	589	471	24
	GCCAGG	519	405	23

FIG. 7.13 – SMILE : inférence d'une boîte sur G_c.
Quorum 40%, longueur minimum 6 pour chaque boîte, une substitution. Les résultats ont été regroupés (à la main) par familles. Les troisième et quatrième colonnes indiquent le nombre de séquences concernées par un modèle dans le jeu original et dans le jeu shufflé (en moyenne).

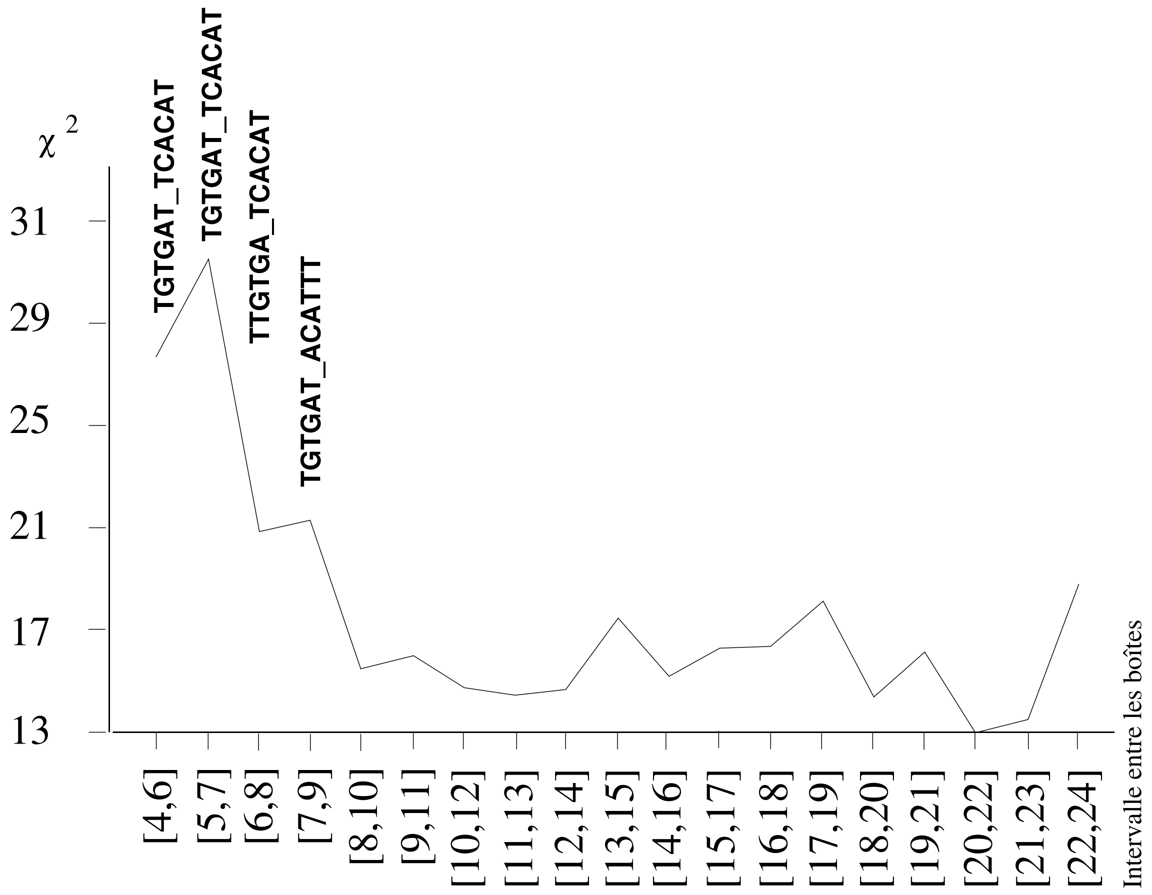

FIG. 7.14 – SMILE : inférence de deux boîtes sur G_c, par balayage de l'intervalle $[4;24]$ avec $\delta = 1$.

Quorum 2%, boîtes de longueur 6 ou 7 et une longueur totale de 13, une substitution globale. Le χ^2 du modèle le plus significatif est indiqué pour chaque intervalle.

B.subtilis
Nous utilisons le jeu G_s qui, en mémoire sous forme d'arbre des suffixes, occupe 21Mo. Nous réalisons une première inférence similaire à celle faite pour le jeu G_c (quorum 40%, longueur minimum 6, une substitution). La simulation réalise mille *shufflings*. On obtient 3 807 modèles en 32mn 2sec (dont une demi-seconde d'extraction), les résultats sont présentés figure 7.15.

Famille	Modèle	Nb seq	Nb seq shff	χ^2
1	TATGATA	627	407	91
	TATCATA	615	403	84
	TATAATAA	445	277	58
	TTATTATA	439	273	57
2	ATGATAA	617	477	36
	ATGAGAA	500	377	29
	TGAGAAA	520	417	19
3	TGACAAA	510	405	21
4	CCTTTTT	638	413	95
	CCTTTTC	496	291	84
	CTCTTTT	600	391	81
	CTTTTCT	613	410	77
	CTTTTTC	652	451	76
5	CCTGCT	478	339	38
	CCCCTT	488	349	37
	CTCCTT	644	512	32
	TCTCCT	609	477	32
	CCGCTT	484	359	30
6	AAAAAGG	629	418	83
	AAAAGAG	599	393	79
	GAAAAGG	496	301	75
	GAAAAAG	651	450	75
	GAAAAGA	604	430	56

FIG. 7.15 – SMILE : inférence d'une boîte sur G_s.
Quorum 40%, longueur minimum 6 pour chaque boîte, une substitution. Les résultats ont été regroupés (à la main) par familles.

Ces résultats recueillent des valeurs de signification statistique nettement plus élevées que dans le cas de *E.coli*. Les familles 1 et 3 semblent approcher les boîtes du consensus connu `TTGACAx[15 ;19]TATAATA`. Les familles 4 et 6 coïncident avec des parties du troisième motif obtenu par MEME (non présenté précédemment, voir [Vanet et al., 1999]) pour lesquels il reste à déterminer s'ils ont une fonction biologique réelle.

Nous réalisons ensuite une inférence de modèles structurés afin de trouver une éventuelle association des deux boîtes observées précédemment qui formeraient le consensus attendu. Nous effectuons une inférence avec un quorum faible (4%) par balayage des intervalles de rayon 1 se trouvant dans l'intervalle $[4; 24]$.

On infère ainsi 8035 modèles en 4h 50mn (dont 40mn pour l'extraction). La simulation est réalisée par cent *shufflings*. Les résultats sont présentés figure 7.16.

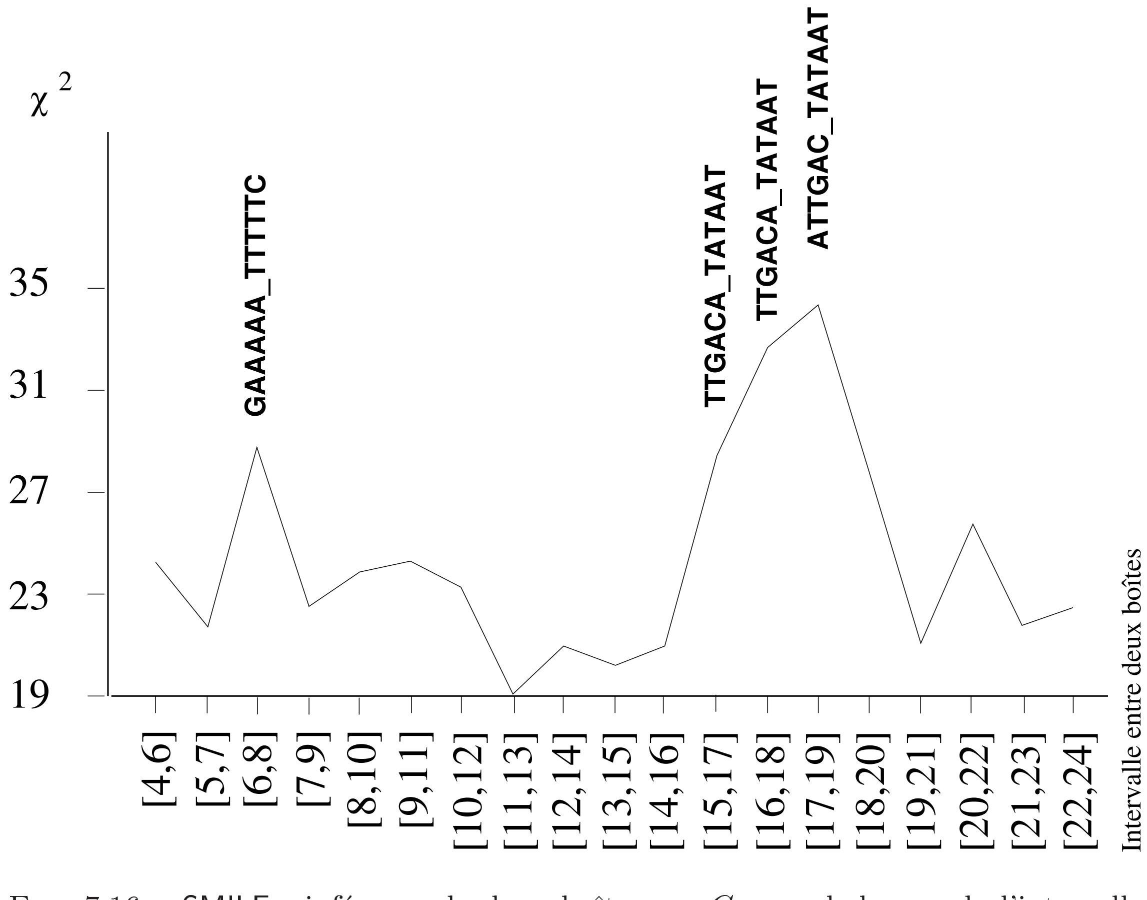

FIG. 7.16 – SMILE : inférence de deux boîtes sur G_s, par balayage de l'intervalle $[4; 24]$ avec $\delta = 1$.
Quorum 4%, boîtes de longueur 6 ou 7 et une longueur totale de 13, une substitution globale.

Le consensus attendu est nettement détectable sur l'intervalle [15 ;19], avec parfois un A bien conservé en amont de la première boîte. Cette inférence mérite d'être reitérée avec des critères plus souples pour rechercher un consensus étendu. On observe également un modèle $GAAAAAx[6;8]TTTTTC$, palindromique, composé des modèles simples des familles 4 et 6 de la première inférence, qui sont cette fois nettement mis en avant. Comme nous le disions précédemment, sa fonction biologique n'est pas connue.

Les résultats obtenus ici sont non seulement positifs sur le plan biologique, mais cautionnent également les choix algorithmiques que nous avons faits. Sur un tel jeu de séquences, issu directement d'un génome, les outils auxquels nous avons comparé SMILE ne sont pas capables de pratiquer une inférence concluante. Par rapport aux approches d'optimisation, SMILE semble faire preuve d'une meilleure capacité d'inférence dans des conditions difficiles.

L'approche combinatoire récente de [Eskin and Pevzner, 2001] utilise des techniques d'extraction de motifs simples inspirées comme les nôtres de [Sagot, 1998]. Ils traitent également le problème des motifs structurés composés de deux boîtes. Pour ce faire, ils génèrent tous les couples possibles de modèles simples obtenus lors d'une première extraction, à la recherche d'une association obtenant un bon score avec un *spacer* de longueur fixe. Cette méthode (qui est similaire à l'approche naïve abordée dans la section 5.2.1.1) étant très coûteuse, les auteurs choisissent de n'associer que les motifs simples obtenant les meilleurs scores (le nombre de motifs retenus est variable). L'algorithme est donc exact pour l'extraction de motifs simples mais heuristique quand il associe deux motifs.

Or, dans les expériences précédentes, lors de l'inférence de modèles simples avec SMILE, le modèle TTGACA est apparu en 1724$^{\text{ème}}$ position. Ce type d'exemple nous amène à penser que l'approche développée par Eskin *et al.* est discutable : une fois de plus, rien ne prouve qu'un modèle structuré statistiquement significatif soit nécessairement composé de motifs qui, pris isolément, soient statistiquement significatifs également.

La représentation des modèles structurés semble être pertinente dans ce cas, puisque l'association TTGACA_TATAAT, apparaissant en première position lors de l'inférence de modèles structurés, se compose de deux boîtes qui n'apparaissaient telles quelles qu'aux 624$^{\text{ème}}$ et 1724$^{\text{ème}}$ positions lors de l'inférence de modèles simples.

H.pylori

Nous utilisons ici le jeu G_p, qui génère un arbre des suffixes de 8Mo. L'inférence de modèles de longueur minimum 6 avec une substitution et un quorum de 40% produit 3561 modèles en 13mn 20sec (dont 0.27sec d'extraction). La simulation est réalisée par mille *shufflings*. Les résultats sont présentés figure 7.17. Les scores obtenus sont assez faibles. La première famille pourraît néanmoins correspondre à une TATA-box.

Famille	Modèle	Nb seq	Nb seq shff	χ^2
	TATGATA	199	101	62
	TATCATA	197	99	61
1	TATAATC	209	119	52
	GTATTAT	199	116	44
	GATTATA	203	122	42
	GGTTTTA	196	132	26
	GGGTTTT	144	85	23
2	ATTTTAGC	125	70	23
	GCTTAAAA	140	85	20
	CCTTAAAA	129	77	19
	TTTTAAGC	142	87	20
3	ATTTTAAG	170	115	19
	TTTTAAGG	134	84	17
	AGCGCT	160	89	34
4	ACGCTA	172	112	23
	ACCGCT	130	75	22
	AACGCTT	133	80	20
	TAGCGT	174	107	29
	AGCGTG	126	68	25
5	AAGCGTT	134	76	24
	AGCGTTT	131	77	21
	AGCGGT	127	76	19
6	AAAACCC	138	82	21

FIG. 7.17 – SMILE : inférence d'une boîte sur G_p.
Quorum 40%, longueur minimum 6 pour chaque boîte, une substitution. Les résultats ont été regroupés (à la main) par familles.

Nous réalisons ensuite, comme précédemment, une inférence de modèles structurés avec un quorum de 6%, en permettant une substitution, par balayage sur l'intervalle $[4; 25]$. Les résultats sont évalués par une simulation de cent *shufflings*. On obtient 57 718 modèles en un peu plus de 6h (dont 6mn d'extraction). La figure 7.18 présente les résultats obtenus. Deux

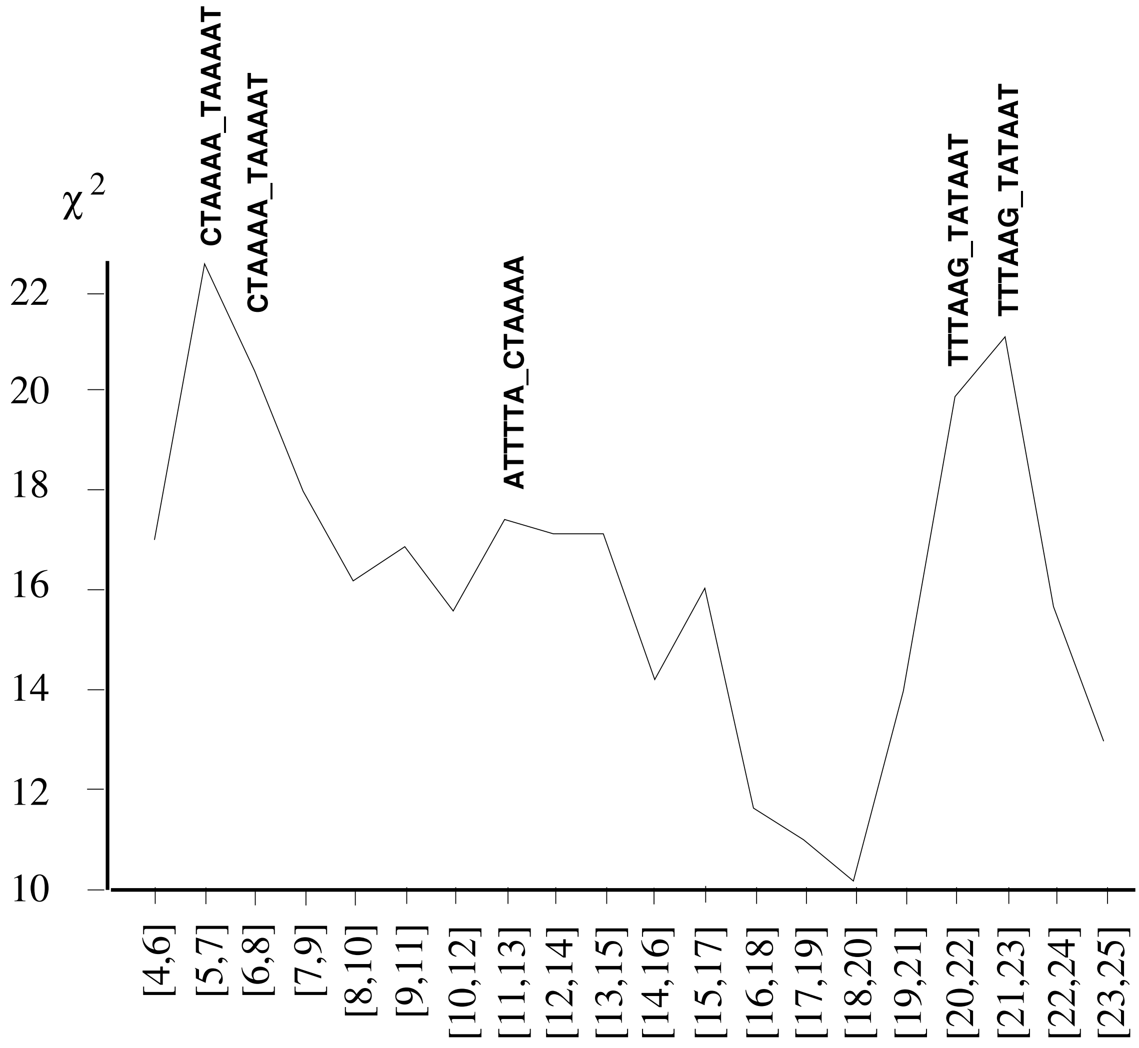

FIG. 7.18 – SMILE : inférence de deux boîtes sur G_p, par balayage de l'intervalle $[4; 25]$ avec $\delta = 1$.
Quorum 6%, boîtes de longueur 6 ou 7, longueur totale 13, une substitution globale.

modèles structurés se détachent nettement, et l'un d'eux ($TTTAAGx[20; 23]TATAAT$) pourrait bien être le site de fixation recherché pour le facteur σ^{80}. Les résultats obtenus lors de l'inférence de modèles simples laissaient penser que celle qui nous apparaît maintenant comme la première boîte d'un modèle structuré pourrait être étendue à gauche, et que la TATA-box pourraît aussi être étendue moyennant l'usage d'un joker. Nous réalisons donc une dernière inférence de deux boîtes de longueur 7 sur l'intervalle [20; 23] dans les mêmes conditions, en autorisant un joker dans la seconde boîte. L'inférence produit 4293 modèles en 14mn 32sec. Les dix meilleurs modèles obtenus sont présentés dans la figure 7.19. Selon qu'on se fie au χ^2 ou au z-score, deux consensus sont nettement mis en

Model	%right	#right	%shfl.	#shfl.	Var.	Chi2	Z-score
TTTTAAG_GTATNAT	5.52%	17	0.49%	1.52	1.16	13.34	13.36
ATTATAG_NTTAAAA	6.82%	21	1.06%	3.26	1.64	13.50	10.84
TTTTAAG_CTANAAT	6.17%	19	0.69%	2.12	1.62	13.97	10.45
TTTTAAG_NTATAAT	8.12%	25	1.26%	3.89	2.08	16.18	10.16
TATTATA_GNTAAAA	5.52%	17	0.76%	2.35	1.47	11.45	9.95
ATTATAC_NTTAAAA	5.84%	18	0.90%	2.76	1.60	11.58	9.55
TATTATA_NCTTAAA	5.52%	17	0.74%	2.29	1.57	11.58	9.40
AATTATA_CTTAANA	5.52%	17	0.88%	2.72	1.54	10.68	9.25
GTTTTTA_GTTANAA	5.52%	17	0.74%	2.28	1.60	11.60	9.19
TATTCTA_NTTAAAA	6.17%	19	1.09%	3.36	1.74	11.35	9.00
...							

FIG. 7.19 – SMILE : inférence de deux boîtes étendues sur G_p, avec l'intervalle [20; 23].
Quorum 6%, longueurs 7 et 7, une substitution globale et un joker dans la seconde boîte.

avant : respectivement `TTTTAAGx[20 ;23]NTATAAT` et `TTTTAAGx[20 ;23]GTATNAT`. Dans les deux cas, la première boîte `TTTTAAG` semble pertinente. La forte couverture (plus de 8%) du premier consensus laisse à penser qu'il est le plus pertinent.

Ces résultats sont très importants pour nous, pour deux raisons :
- nous les avons obtenus avec moins d'informations sur les consensus à extraire que dans le cas des jeux G_c et G_s. Aussi montrent-ils la capacité de SMILE à inférer des sites dont on connaît peu de caractéristiques ;
- les promoteurs du facteur σ^{80} de *H.pylori* n'étant pas aussi bien connus que ceux des deux autres bactéries que nous avons manipulées, les résultats obtenus ici pourraient fournir de sérieuses pistes de recherche aux biologistes travaillant à sa détection. On trouvera de plus amples interprétations biologiques dans [Vanet et al., 2000].

SMILE fait à nouveau preuve dans cette expérience d'une capacité de détection supérieure à celle des autres outils utilisés, parvenant à isoler nettement plusieurs consensus faiblement représentés, là où MEME et Motifs Sampler ne parviennent à mettre en avant aucun consensus.

7.2.3 Discussion

Sur les trois jeux de données manipulés, SMILE fait preuve d'une capacité de détection supérieure à celle des autres outils, s'appuyant surtout sur une bonne spécificité. Lors de certaines extractions de modèles structurés, nous avons même observé des cas où les critères structurels étaient suffisants pour mettre en évidence des sites particuliers, sans avoir besoin de recourir à une évaluation.

Chez *B.subtilis* et *H.pylori*, on parvient à isoler nettement des modèles structurés composés de deux boîtes, qui correspondent aux consensus connus, tandis que MEME et Motifs Sampler, quand ils parviennent à isoler des consensus significatifs, n'obtiennent aucun résultat qui semble s'approcher de ceux attendus. La représentation sous forme de modèles structurés semble donc être judicieuse dans ces expériences. L'approche que nous avons développée s'avère aussi être moins sensible au bruit : les approches d'optimisation présentées ne parviennent pas à détecter des signaux ayant une couverture très réduite. Dans [Vanet et al., 1999], nous faisons également état de la sensibilité de MEME aux variations, même très faibles, dans le jeu de données.

La spécificité qu'apportent les modèles structurés nous a permis de mettre en évidence un couple de sites chez *H.pylori*, qui pourrait correspondre à la séquence promotrice associée au facteur σ^{80}. Notre outil d'inférence joue ici pleinement son rôle, proposant à l'issu d'expériences *in silico* sur des données brutes contenant des signaux inconnus des résultats qui peuvent permettre de faciliter des expérimentations biologiques. À ce jour, ces résultats n'ont à notre connaissance pas été confirmés ou infirmés par de telles expériences. L'article de revue récent de [Bjorkholm et al., 2001] sur *H.pylori* présente l'état des connaissances actuelles sur cet organisme, et, citant nos travaux, permet de situer les résultats que nous avons obtenus dans ce contexte.

Chez *E.coli*, les sites à détecter sont visiblement trop faibles pour être isolés par aucun des outils utilisés, dont SMILE. Nous en concluons qu'il manque certains critères à nos modèles pour permettre la détection des signaux présents chez cet organisme. Plusieurs observations pourraient de plus justifier ces résultats. Il a en effet été observé ([Mulder et al., 1997]) que les promoteurs d'organismes Gram-positifs comme *B.subtilis* ont des consensus plus stricts que ceux de *E.coli*. Les sites de régulation associés au facteur CRP viennent de plus participer à l'initiation de la transcription de certains gènes chez cet organisme, ce qui laisse à penser que certains des sites en -10 et -35 subissent une pression de sélection moins forte que ceux de *B.subtilis* ou *H.pylori*.

D'un point de vue algorithmique, les résultats présentés dans ce chapitre appellent de nombreuses remarques. On peut d'abord constater que les temps d'exécution des inférences sont assez raisonnables, comparativement à ceux d'une heuristique comme celle employée dans MEME. Dans les phases d'investigation, inférant des modèles simples, les extractions sont très rapides. L'extraction de modèles structurés est plus coûteuse, et il est plus difficile de jouer avec les paramètres sans quelques connaissances du modèle à extraire. À ce sujet, les deltas, qui contribuent fortement à la souplesse des modèles à extraire, deviennent coûteux quand on élargit l'intervalle de saut. Cependant, leur implantation doit faire l'objet de travaux ultérieurs car elle peut probablement être optimisée.

On note également que les simulations peuvent devenir très gourmandes en temps quand on extrait un grand nombre de modèles, surtout si ce sont des modèles structurés. Nous avons bon espoir de pouvoir les remplacer à court terme par les méthodes probabilistes présentées dans le chapitre précédent qui, au moins pour des modèles composés de deux boîtes, offrent des résultats similaires pour un coût bien moindre.

Parmi les résultats présentés dans ce chapitre, nous observons fréquemment des corrélations entre les modèles produits. Dans certains cas, les meilleurs modèles obtenus sont des décalages d'un même consensus plus large, ou bien certaines positions semblent susceptibles de recevoir différents symboles. Ces remarques nous amènent à imaginer la construction de méta-modèles à partir des modèles obtenus. Il ne s'agit pas ici d'un simple assemblage des consensus, mais d'une analyse des occurrences de plusieurs modèles afin de déterminer si un autre modèle pourrait les réunir. Cette notion de superposition de modèles sera développée dans la conclusion.

7.3 Applications en cours

Au hasard des rencontres, nous avons été amenés à collaborer avec plusieurs personnes, sur l'étude des génomes de différents organismes :

- *Mycobacterium tuberculosis* et *Mycobacterium leprae*. Hormis des études séparées du même type que celles présentées dans ce chapitre, nous avons aussi étudié ces deux organismes de façon comparée. En effet, il semblerait que *M.leprae* ait perdu un nombre considérable de gènes par rapport à *M.tuberculosis*, certains étant devenus des pseudo-gènes et d'autres ayant disparus (ou étant trop dégénérés pour être associés à des gènes de *M.tuberculosis*). Nous avons donc essayé de détecter les sites de fixation des facteurs de transcription qui n'apparaissent plus que dans l'un de ces deux organismes en nous appuyant sur cette divergence. L'évaluation par la méthode « contre » a été developpée dans ce cadre. Ces travaux ont été effectués en collaboration avec Anne Cariou (stagiaire à l'Institut Pasteur, venant de l'école vétérinaire de Maison-Alfort) et Katja Schuerer (SIS - Institut Pasteur) ;
- *Synechocystis* PCC 6803. Cette cyanobactérie a la particularité d'être une bactérie qui a des caractéristiques n'étant pas toujours très « bactériennes ». C'est donc pour nous un sujet d'études ayant une complexité intermédiaire entre procaryotes et eucaryotes. Ces travaux sont réalisées en collaboration avec Nicole Tandeau de Marsac (Institut Pasteur), Hans Geiselmann (Université Joseph Fourier de Grenoble), Anne Morgat et Hilde de Jong (INRIA Rhône-Alpes - Projet HELIX) ;
- nous collaborons avec Élodie Guillaume, Claude Thermes, et Yves d'Aubenton-Caraffa (du Centre de Génétique Moléculaire de Gif-sur-Yvette) qui travaillent à la détection de séquences *enhancer* impliquées dans l'épissage chez les vertébrés, et en particulier chez l'homme ;
- nous collaborons avec Ingrid Lafontaine (Institut Pasteur), qui a développé une mé-

thode de modélisation moléculaire (ADAPT, voir [Lafontaine and Lavery, 2001]). Cette méthode permet d'étudier l'influence de la séquence primaire de l'ADN sur sa structure. Elle détermine ainsi certaines zones d'une séquence plus aptes à subir des interactions ou des déformations données. Ses expériences sur des séquences génomiques humaines lui ont ainsi permis de détecter des promoteurs associés à la TBP (TATA-*box Binding Protein*) avec précision. SMILE, appliqué à la détection de ces promoteurs, a fait preuve d'efficacité en parvenant à isoler ces promoteurs dans un jeu de séquences conséquent (644Ko).

Conclusions et perspectives

La destination du chercheur dépend de la route qu'il suit.
Ibn al-'Arabi. (1165-1240).

Les travaux initiés durant cette thèse nous paraissent difficiles à conclure à ce jour. Il serait en effet bien prétentieux de prétendre avoir développé des algorithmes et outils permettant l'inférence de sites de fixation. Tout au plus avons-nous développé par nos travaux de nouvelles notions, ainsi que les algorithmes et outils associés, permettant de détecter certains sites avec, dans certains cas, un peu plus d'acuité que ceux existants. Mais le problème de l'inférence de sites de fixation reste extrêmement complexe, et les applications que nous avons menées, si elles se sont soldées par certains résultats concluants, nous ont également fait découvrir l'ampleur de la difficulté à comprendre les lois qui régissent les mécanismes de régulation des gènes. Ainsi, l'effervescence apportée par la détection de signaux que d'autres algorithmes ne parviennent pas à isoler est-elle bien souvent pondérée par le constat de notre impuissance à en détecter d'autres.

Par ce travail, nous pensons avoir contribué à montrer que l'on peut développer des algorithmes et outils exacts et performants intégrant des critères complexes. Nos travaux ont également permis de montrer que la représentation développée, les modèles structurés, possède un rapport sensibilité/coût concluant, qui permet l'inférence de signaux très faibles dans de grands jeux de séquences, là où les matrices couramment employées et les algorithmes d'optimisation associés sont souvent submergés par le bruit. À ce titre, nous pensons surtout avoir apporté un peu de diversité parmi les outils d'inférence existants (et tout biologiste sait que la diversité est un des moteurs de l'évolution !), essentiellement orientés vers l'utilisation de méthodes et de représentations dont le quasi-monopole mérite, à notre avis, d'être remis en question.

Il semble donc évident que nos travaux dans ce domaine ne se terminent pas avec cette thèse. Les questions qui restent ouvertes en matière d'inférence de sites de fixation sont en effet plus nombreuses et variées que jamais. Classons-les dans deux catégories :

- les questions de définition des représentations et des problèmes à résoudre. Les réponses à ces questions fournissent le lien entre biologie et algorithmique, elles ne peuvent donc pas être considérées indépendamment d'un de ces deux domaines ;
- les questions algorithmiques pures, qui se posent après avoir déterminé des solutions aux questions précédentes. Elles sont théoriquement indépendantes du problème bio-

logique initial. En pratique, les problèmes induits par les définitions choisies sont souvent trop complexes pour être résolus tels quels. Ils nécessitent souvent de faire certains compromis sur ces définitions, afin d'obtenir des problèmes algorithmiques moins complexes mais conservant une pertinence biologique.

Perspectives concernant les définitions et représentations

Une des principales problématiques auxquelles nous sommes confrontés concerne l'amélioration de la représentation utilisée pour modéliser les sites de fixation. Nos travaux ont permis de développer une représentation prenant en compte des associations de sites, les modèles structurés. Cette spécificité est à ce jour peu utilisée dans des algorithmes d'inférence. Plusieurs publications récentes, prenant en compte de telles associations, prouvent cependant un intérêt croissant pour l'intégration de telles associations dans les modélisations de sites de fixation (voir entre autres [Liu et al., 2001b], [Guha-Thakurta and Stormo, 2001] qui sont des approches d'optimisation, et [Sinha, 2001] qui passe en revue certaines représentations et approches). À notre connaissance, aucune approche exacte ne traite l'inférence de modèles structurés composés de deux boîtes ou plus, en autorisant les substitutions.

Les applications que nous avons présentées montrent bien que la prise en compte de modèles structurés améliore dans certains cas la sensibilité et la spécificité, et autorise la détection de sites qui n'auraient pas pu être détectés indépendamment. Force est pourtant de constater qu'il manque aux modèles structurés des caractéristiques primordiales pour mettre en évidence certains sites, comme les promoteurs associés au facteur σ^{70} de *E. coli*.

Dans le but d'améliorer la sensibilité des algorithmes de détection, nous considérons deux directions à prendre :

- laisser le problème d'inférence dans les termes où nous l'avons posé ici, c'est-à-dire ramené au problème de détection de motifs communs approchés dans un jeu de séquences primaires, et considérer que les représentations sont la source des incapacités à détecter certains sites. Il faudrait alors envisager d'augmenter le pouvoir de description des modèles structurés, en y incluant de nouvelles caractéristiques qui pourraient être pertinentes. Par exemple, gérer des corrélations entre plusieurs positions, ou entre boîtes (palindromes [5], répétitions), autoriser des insertions ou délétions de symboles, voire de boîtes complètes. Dans le cadre d'une inférence d'autres signaux dans d'autres types de séquences (nous pensons particulièrement aux séquences protéiques), il faudrait envisager la prise en compte d'autres caractéristiques propres. La complexité des caractéristiques accumulées pour décrire un signal nous amène à imaginer le développement d'un langage qui permettrait de les exprimer efficacement et simplement ;
- revoir la définition du problème d'inférence qui, telle que nous l'avons posée, n'offre peut-être pas assez d'informations pour détecter tous les types de sites de fixation, et ce quelle que soit la représentation employée. En ce sens, nous devrions envisager de

[5] dans les toutes premières versions de SMILE traitant les modèles répétés dans une séquence, nous avions intégré la gestion éventuelle de palindromes. Celle-ci n'a pas été propagée aux versions ultérieures, mais son implantation ne pose aucune difficulté majeure.

prendre en compte des données supplémentaires sur les séquences d'ADN utilisées, toujours sous réserve que ces données soient disponibles, ce qui reste rarement le cas. Ainsi, la conformation spatiale de l'ADN (voir l'article récent de [Liu et al., 2001a]) et ses propriétés physico-chimiques pourraient être intégrées dans les méthodes de comparaison utilisées. Pour peu que la conformation spatiale d'un facteur de transcription soit connue également, on pourrait la prendre en compte pour déduire ses interactions possibles avec l'ADN (le nombre de bases entre deux grands sillons successifs, ou l'angle de rotation entre deux nucléotides successifs par exemple) afin de réduire l'espace de recherche. Les données d'expression, de plus en plus souvent disponibles, pourraient, elles, permettre de réduire les jeux de données employés et d'espérer de meilleures couvertures des sites à inférer. L'utilisation de plusieurs jeux de séquences contenant des gènes orthologues (approche d'inférence phylogénétique) est également une solution qui pourrait aider à mettre en évidence certains sites. La prise en compte de ces nouvelles informations implique de développer de nouvelles tactiques d'inférence, ainsi que des algorithmes adaptés.

Ces différentes solutions doivent être considérées du point de vue de leur pertinence biologique et du coût algorithmique qu'elles impliquent.

Concernant la représentation des sites de fixation, le traitement des résultats obtenus lors d'une inférencee est un autre point qui, à notre avis, nécessite d'être exploré. À cette étape d'une inférence, la priorité n'est plus de séparer les bons résultats des mauvais, mais de réunir les meilleurs obtenus pour en construire de nouveaux plus pertinents. Nous discuterons plus loin de l'idée qui consisterait à utiliser les modèles structurés de SMILE comme des alignements de départ pour un algorithme d'optimisation. Avant d'avoir recours à d'autres algorithmes, nous pouvons considérer que l'ensemble des modèles et occurrences produits au terme d'une inférence contient des informations qui permettraient à elles seules de construire des modèles plus pertinents. Nous pensons que de nombreux travaux restent à faire à partir de ces résultats. Nous avons en effet constaté lors de nos applications que les modèles obtenus étaient parfois de simples décalages les uns des autres, laissant supposer la présence d'un consensus plus large. On observe également parfois plusieurs modèles proches, où seule une position varie, qui semble donc être plus faible et nécessiter l'emploi d'un symbole dégénéré. Les traitements à partir des modèles obtenus et de leurs occurrences, en vue de construire des méta-modèles, sont actuellement effectués à la main. Il serait intéressant de travailler à l'élaboration de méthodes visant à automatiser cette tâche, ou à intégrer ce traitement dans les algorithmes d'inférence.

En aval de l'inférence, des travaux récents présentés dans [Blanchette and Sinha, 2001] réalisent ce type de traitement par une approche statistique : ils cherchent à dégager des consensus plus nets en essayant d'éliminer ceux qui ne sont présents qu'à cause de leur proximité avec un consensus fort.

Nous nous intéressons plutôt à l'intégration de propriétés algorithmiques supplémentaires dans les modèles, qui permettraient de ne générer que de tels méta-modèles. Dans [Parida et al., 2000], des propriétés sont introduites qui permettent de définir une *base de motifs* pour l'ensemble des motifs de la séquence. Il est démontré que les éléments de cette base sont en nombre linéaire en la longueur de la séquence utilisée. Les algorithmes d'extraction des motifs composant cette base sont évidemment plus coûteux que ceux visant à inférer des motifs classiques. L'application de telles notions (qu'il reste à étendre à la gestion de substitutions et d'associations de boîtes) aux modèles que nous utilisons, et la recherche d'algorithmes associés efficaces serait une direction vers laquelle nous pourrions orienter nos travaux.

Observant les caractéristiques, plus souples encore, des sites de fixation eucaryotes (distance très variables entre boîtes, insertion et délétion de boîtes complètes, interversions de boîtes), nous ressentons la nécessité d'étendre encore le voisinage d'un modèle en autorisant de nouvelles altérations. Nous pourrions imaginer une description des séquences initiales par les modèles structurés obtenus lors d'une première inférence classique, qui serait suivie d'une nouvelle « super-inférence » visant à assembler ces modèles, devenus « méta-symboles », en groupes de modèles structurés. Bien qu'augmentant encore l'espace des solutions à considérer de manière conséquente, il serait cependant vivement souhaitable de travailler à intégrer le traitement de ces assemblages à l'intérieur même des algorithmes d'extraction, comme nous l'avons réalisé pour les modèles structurés, plutôt qu'en post-traitement.

L'évaluation de la signification des modèles que nous obtenons reste également un problème important à traiter. Nous avons vu que les méthodes de simulation s'avèrent très coûteuses quand on considère des modèles complexes. Les formules de probabilité d'apparition de la première occurrence d'un modèle développées dans [Régnier and Szpankowski, 1997] et [Robin et al., 2001] pourraient offrir des solutions particulièrement efficaces.

Les questions concernant les mesures à utiliser et le contexte à prendre en compte pour obtenir une estimation valable de l'activité d'un site restent cependant d'actualité. Ici aussi, une voie se dessine qui consisterait à considérer des critères supplémentaires pour réaliser cette estimation. Il serait en effet intéressant de s'interroger sur le « champ de vision » que possède un facteur de transcription destiné à se fixer aux sites que nous recherchons : est-ce que la conformation de l'ADN favorise certaines zones plutôt que d'autres lors de la reconnaissance des sites ? Comment caractériser ces zones ? Peut-on incorporer dans nos méthodes d'évaluation, voire d'inférence, de telles informations ? Ces questions nécessitent évidemment d'être discutées avec des biologistes. Elles se heurtent malheureusement souvent aux limites actuelles des connaissances des mécanismes de reconnaissance des sites de fixation.

Perspectives algorithmiques

L'optimisation des algorithmes d'inférence est d'importance : elle sert indirectement la sensibilité aux sites à détecter puisqu'elle permet de visiter des espaces de solutions plus vastes en utilisant des représentations plus complexes. Le but de l'algorithme de Marie-France Sagot, sur lequel nous nous sommes appuyés, était de réduire le parcours de l'espace des solutions tout en conservant un caractère exact. Le lemme sur lequel se fonde cet algorithme permet d'éviter de visiter les modèles dont un préfixe ne respecte pas le quorum requis. La vérification de cette propriété est facilitée par la structure récursive de l'algorithme. Cette propriété n'est cependant pas utilisée pour éviter de visiter un modèle dont un suffixe ne recueillerait plus le quorum. Les extensions maximales que nous avons introduites pallient à ce problème, avec certaines contraintes néanmoins : la méthode utilisée n'est pas optimale, et nous ne l'utilisons que pour des suffixes de taille limitée, au risque de « casser » la linéarité de l'arbre des suffixes. Concernant cet algorithme et les nôtres qui en découlent, nous pensons donc que plusieurs points importants méritent d'être étudiés plus avant :

- d'un point de vue théorique, l'obtention de bornes pour la complexité de nos algorithmes dotés des extensions maximales serait souhaitable. En pratique, il faudrait également implanter cet algorithme afin d'estimer les coût et gain réels de ces extensions maximales telles que nous les avons définies ;
- sans modifier les algorithmes, il nous faut chercher une structure de données adaptée à l'utilisation des extensions maximales, qui permettrait de minimiser leur coût en espace ;
- avec les extensions maximales, nous recherchons une optimisation potentielle chez les suffixes des modèles. Mais, contrairement au lemme appliqué aux préfixes des modèles dans l'algorithme original, cette optimisation n'est pas exacte : il se peut qu'un modèle ayant un suffixe ne respectant pas le quorum soit tout de même étendu, faute d'avoir pu obtenir à temps suffisamment d'informations sur ce suffixe. Cela est dû au fait que l'algorithme réalise un parcours préfixe de l'arbre virtuel des modèles. Il serait très intéressant d'explorer d'autres algorithmes et structures de données permettant un parcours en largeur afin de réduire encore l'exploration de l'espace des solutions. Le compromis est ici de conserver un algorithme linéaire en la longueur des séquences et d'obtenir une complexité en espace raisonnable.

Concernant les algorithmes que nous avons développés, plusieurs travaux restent à réaliser à ce jour. L'algorithme 2 a été implanté par un étudiant en thèse à Nantes, avec qui nous souhaiterions déterminer si les complexités observées en pratique sont effectivement plus intéressantes que celles de l'algorithme 1. Dans nos deux algorithmes d'extraction de modèles structurés, il nous semble que nombre d'optimisations sont possibles permettant d'économiser certains parcours, en particulier lors des sauts utilisant des deltas. Ces questions doivent être explorées.

Nos algorithmes et outils disposent d'atouts de taille : ils sont exacts, ainsi tout résultat obtenu peut être interprété, et une absence de résultat est aussi un résultat puisqu'on peut garantir qu'aucune solution n'existe au problème tel qu'il est posé. Ils permettent l'inférence de modèles complexes et faiblement représentés, dans de grands jeux de données. Les quelques applications présentées montrent bien le positionnement de SMILE parmi les algorithmes existants : nous ne prétendons pas supplanter des outils d'optimisation comme MEME ou Motifs Sampler, mais permettre une approche différente du problème. Selon nous, les approches combinatoires et d'optimisation sont complémentaires, et gagneraient à être utilisées de façon coopérative. Par exemple, lors d'une inférence en terrain inconnu, il serait intéressant de s'aider d'une approche d'optimisation réalisant un parcours partiel de l'espace des solutions afin de fournir des paramètres pertinents à notre algorithme combinatoire. En effet, malgré la possibilité d'exprimer des fourchettes de longueurs pour les modèles et les *spacers*, ainsi que des deltas, certains paramètres tel le quorum sont délicats à estimer lors d'inférences sans connaissances *a priori*.

Une autre interaction avec des algorithmes d'optimisation nous semble pertinente à mettre en oeuvre : une fois un consensus inféré à l'aide de SMILE, nous effectuons en général plusieurs inférences complémentaires visant à affiner ce consensus, à l'élargir ou à fusionner plusieurs modèles se chevauchant. À ce stade, il serait très intéressant de pouvoir fournir les occurrences d'un modèle à un algorithme d'optimisation qui s'en servirait comme alignement de départ pour tenter de converger vers un meilleur consensus. Cette idée, qui nous a été suggérée par un arbitre lors de la soumission d'un article, mérite d'être explorée. Elle permettrait en fait de compenser les faiblesses de chacune des deux approches par les atouts de l'autre. Le problème de l'évaluation trop simple de la signification des motifs manipulés dans les approches EM ou Gibbs classiques semble trouver des solutions dans des algorithmes d'optimisation récents (comme [Guha-Thakurta and Stormo, 2001] ou [Thijs et al., 2001b]), qu'il pourrait s'avérer fructueux de faire interagir avec SMILE. Pour faire coopérer nos algorithmes avec une approche d'optimisation, il faudrait que nous puissions tabler sur une mesure de signification commune, sans quoi, même avec un alignement de départ issu d'une inférence réalisée avec SMILE, un algorithme d'optimisation pourrait ne pas converger dans la direction voulue.

Annexe A

Expériences sur les conditions de simulation

Nous avons utilisé les résultats d'extractions de modèles simples obtenus sur le jeu G_s (section 7.2.2.3, page 188) pour réaliser quelques expériences qui visaient à observer l'influence de la taille des k-mers conservés lors des *shufflings* de la simulation sur la signification statistique des modèles extraits.

Nous pratiquons généralement des conservations en di-nucléotides, cette longueur semblant être celle qui permet de mettre le mieux en évidence les motifs détectés. Dans nos expérimentations, nous avons fait varier cette longueur de 1 à 4 afin d'observer son influence sur les tests statistiques appliqués lors de l'évaluation.

La figure A.1 donne les scores et les positions des 10 meilleurs modèles recueillis au terme de mille *shufflings* en mono, di, tri et quadri-nucléotides, lors de l'inférence de modèles simples de longueur minimum 6 avec une substitution et un quorum de 40%, dans le jeu G_s. L'augmentation de la longueur des k-mers conservés entraîne nécessairement une diminution des mesures statistiques obtenues : on renforce les contraintes imposées aux séquences shufflées, donc le nombre de séquences possibles qui les respecte diminue et les chances de construire la séquence initiale augmentent. Quand les séquences shufflées se rapprochent des séquences originales, les modèles relevés obtiennent des couvertures plus proches des couvertures initiales, et donc une signification statistique plus faible.

La conservation en mono-nucléotides entraîne un classement privilégié des suites de A ou T, tandis que celle en quadri-nucléotides met en avant des modèles composés de A et de T répartis équitablement. Si on considère que les séquences, riches en AT, contiennent naturellement des suites de A ou de T, on peut imaginer que le *shuffling* en mono-nucléotides « casse » suffisamment ces séquences pour qu'on ne retrouve pas autant de ces suites dans les séquences shufflées, ce qui peut leur donner une signification statistique plus importante.

Fait intéressant, le meilleur modèle évalué lors de la simulation conservant les quadri-nucléotides recueille quasiment la même couverture dans les séquence shufflées (34%, non prśenté) que dans les séquences shufflées issues de la simulation conservant les mono-nucléotides, au terme de laquelle il se retrouve à la position 1001. Cette mauvaise position ne semble pas tant due à une forte présence dans les séquences shufflées qu'au fait qu'il y ait beaucoup de modèles (les suites de A ou de T) qui obtiennent des scores plus élevés.

Il nous semble de toute façon difficile de conclure sur ces résultats. L'augmentation de la longueur des k-mers conservés semble ici mettre radicalement en avant des modèles plus pertinents, mais lors de plusieurs tests sur les résultats d'inférence de modèles structurés, nous n'avons pas observé de bouleversements dans les classements obtenus. L'influence de l'ordre de la conservation des séquences shufflées sur la signification des modèles obtenus doit donc être étudiée plus avant.

Modèle	mono		di		tri		quadri	
	Z-score	Pos.	Z-score	Pos.	Z-score	Pos.	Z-score	Pos.
CTTTTTTT	23.28	**1**	12.89	15	4.30	243	2.38	289
AAAAAGG	22.43	**2**	13.77	3	6.02	74	2.57	293
CTTTTTC	22.14	**3**	13.11	8	5.14	144	2.57	294
CCTTTTT	21.86	**4**	14.58	2	7.05	28	3.23	151
TTTTTTCA	21.30	**5**	10.47	53	2.06	973	1.00	1146
GAAAAAG	21.08	**6**	12.93	13	4.86	172	2.76	243
TTTTTTTC	20.98	**7**	10.81	42	1.86	1075	0.33	1697
AAAAAAAG	20.94	**8**	11.50	26	3.40	461	1.69	662
GAAAAGG	20.81	**9**	13.67	7	6.81	36	3.93	69
GAAAAAAA	20.69	**10**	9.56	97	1.29	1389	-0.02	2023
CCTTTTC	20.62	12	15.13	**1**	7.42	14	4.06	58
CCTTTTT	21.86	4	14.58	**2**	7.05	28	3.23	151
AAAAAGG	22.43	2	13.77	**3**	6.02	74	2.57	293
TATGATA	7.52	614	13.75	**4**	7.77	12	6.34	2
CTTTTCT	18.43	30	13.75	**5**	6.30	57	3.62	99
CTCTTTT	17.96	43	13.70	**6**	7.14	26	4.72	20
GAAAAGG	20.81	9	13.67	**7**	6.81	36	3.93	69
CTTTTTC	22.14	3	13.11	**8**	5.14	144	2.57	294
TCCTTTT	19.76	17	13.10	**9**	5.72	91	2.18	432
AAAAGAG	17.96	41	13.07	**10**	7.42	15	4.78	19
TATAATAA	5.29	1001	11.18	30	9.43	**1**	6.38	1
TTATAATA	4.87	1081	9.92	74	9.34	**2**	5.97	4
TTATTATA	4.95	1071	10.72	44	9.16	**3**	5.82	6
TTACTAT	-0.47	2302	6.53	442	9.00	**4**	4.42	33
TATAATA	4.12	1230	12.37	18	8.82	**5**	5.55	7
TACTATA	-1.39	2581	10.37	56	8.55	**6**	4.90	16
TATTATAA	4.45	1163	9.87	79	8.51	**7**	5.45	9
TATTATA	4.05	1242	11.42	28	8.33	**8**	5.17	12
TATAGTA	-2.22	2808	9.87	80	7.87	**9**	4.28	39
CATTATA	4.99	1061	9.29	112	7.87	**10**	3.99	65
TATAATAA	5.29	1001	11.18	30	9.43	1	6.38	**1**
TATGATA	7.52	614	13.75	4	7.77	12	6.34	**2**
GCATGC	8.66	503	5.90	568	3.79	346	6.31	**3**
TTATAATA	4.87	1081	9.92	74	9.34	2	5.97	**4**
TATCATA	6.54	779	12.93	14	7.01	29	5.85	**5**
TTATTATA	4.95	1071	10.72	44	9.16	3	5.82	**6**
TATAATA	4.12	1230	12.37	18	8.82	5	5.55	**7**
AAAATGG	15.07	111	7.89	261	7.35	20	5.54	**8**
TATTATAA	4.45	1163	9.87	79	8.51	7	5.45	**9**
AAATGAG	10.78	309	6.90	372	6.08	69	5.35	**10**

TAB. A.1 – Comparaison de simulations par *shufflings* avec conservation en mono, di, tri et quadri-nucléotides, sur le jeu G_s.

Les 10 meilleurs modèles pour chaque simulation sont présentés et comparés (en Z-score et en position dans le classement) avec les trois autres séries.

Bibliographie

[Alberts et al., 1995] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., and Watson, J. (1995). *Biologie moléculaire de la cellule*. Flammarion.

[Alder and Roessler, 1972] Alder, H. L. and Roessler, E. B. (1972). *Introduction to probability and statistics*. W.H. Freeman and Company.

[Allali, 2000] Allali, J. (2000). Structures d'indexation : les arbres des facteurs. Rapport de stage de Maîtrise - Univ. Marne-la-Vallée. Paris, France.

[Allali and Sagot, 2002] Allali, J. and Sagot, M.-F. (2002). Factor trees (titre provisoire). En cours de rédaction.

[Apostolico et al., 2000] Apostolico, A., Bock, M. E., Lonardi, S., and Xu, X. (2000). Efficient detection of unusual words. *J. of Comp. Biol.*, 7(1/2) :71–94.

[Arnone and Davidson, 1997] Arnone, M. I. and Davidson, E. (1997). The hardwiring of development : organization and function of genomic regulatory systems. *Development*, 124 :1851–1864.

[Baeza-Yates and Gonnet, 1992] Baeza-Yates, R. and Gonnet, G. H. (1992). A new approach to text searching. *Commun. ACM*, 35 :74–82.

[Bailey, 1995] Bailey, T. L. (1995). *The approximate common substring problem*. PhD thesis, University of California, San Diego, USA.

[Bailey and Elkan, 1994] Bailey, T. L. and Elkan, C. (1994). Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Second International Symposium on Intelligent Systems for Molecular Biology*, pages 28–36, Stanford University, Stanford, CA, USA.

[Bailey and Elkan, 1995a] Bailey, T. L. and Elkan, C. (1995a). ParaMEME : a parallel implementation and a web interface for a DNA and protein motif discovery tool. *Comp. Appl. Biosci.*, 12 :303–310.

[Bailey and Elkan, 1995b] Bailey, T. L. and Elkan, C. (1995b). Unsupervised learning of multiple motifs in biopolymers unsing EM. *Machine Learning*, 21 :51–80.

[Baldi and Brunak, 1998] Baldi, P. and Brunak, S. (1998). *Bioinformatics : the machine learning approach (adaptative computation and machine learning)*. MIT Press.

[Berg and von Hippel, 1987] Berg, O. and von Hippel, P. (1987). Selection of DNA binding sites by regulatory proteins. i. statistical-mechanical theory and application to operators and promoters. *J. Mol. Biol.*, 193 :723–750.

[Berg and von Hippel, 1988] Berg, O. and von Hippel, P. (1988). Selection of DNA binding sites by regulatory proteins. ii. the binding specificity of cyclic AMP receptor protein to recognition sites. *J. Mol. Biol.*, 200 :709–723.

[Bieganski et al., 1994] Bieganski, P., Riedl, J., Carlis, J. V., and Retzel, E. (1994). Generalized suffix trees for biological sequence data : applications and implementations. In *Proc. of the 27th Hawai Int. Conf. on Systems Sci.*, pages 35–44. IEEE Computer Society Press.

[Bjorkholm et al., 2001] Bjorkholm, B., Oh, J., Falk, P., Engstrand, L., and Gordon, J. (2001). Genomics and proteomics converge on helicobacter pylori. *Curr. Opin. Microbiol.*, 4(3) :237–245.

[Blanchette and Sinha, 2001] Blanchette, M. and Sinha, S. (2001). Separating real motifs from their artifacts. *Bioinformatics*, 1 :1–8.

[Boyer and Moore, 1977] Boyer, R. and Moore, J. S. (1977). A fast string searching algorithm. *Commun. ACM*, 20 :(10)762–772.

[Brazma et al., 1998a] Brazma, A., Jonassen, I., Eidhammer, I., and Gilbert, D. (1998a). Approaches to automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5 :277–304.

[Brazma et al., 1997] Brazma, A., Jonassen, I., Vilo, J., and Ukkonen, E. (1997). Data mining for regulatory elements in Yeast genome. In Gaasterlan, T., P.Karp, K.Karplus, C.Ouzounis, C.Sander, and A.Valencia, editors, *Fifth International Symposium on Intelligent Systems for Molecular Biology*, pages 65–74, Halkidiki, Greece. AAAI Press.

[Brazma et al., 1998b] Brazma, A., Jonassen, I., Vilo, J., and Ukkonnen, E. (1998b). Predicting gene regulatory elements in silico on a genomic scale. *Genome research*, 8 :1202–1215.

[Bréhélin and Gascuel, 2000] Bréhélin, L. and Gascuel, O. (2000). Modèles de Markov cachés et apprentissage de séquences. In Prade, H., Jeansoulin, R., and Garbay, C., editors, *Le temps, l'espace et l'évolutif en sciences du traitement de l'information*, pages 407–421. Éditions Cépaduès.

[Bucher, 1990] Bucher, P. (1990). Weight matrix description of four eukaryotic RNA polymerase ii promoter. *J. Mol. Biol.*, 212 :563–578.

[Bucher and Bryan, 1984] Bucher, P. and Bryan, B. (1984). Signal search analysis : a new method to localize and characterize functionnaly important DNA sequences. *Nucleic Acid Res.*, 12 :287–305.

[Bucher et al., 1996] Bucher, P., Fickett, J., and Hatzigeorgiou, A. (1996). Computational analysis of transcriptional regulatory elements : a field in flux. *Comput. Appl. Biosci.*, 12 :361–362.

[Buhler and Tompa, 2001] Buhler, J. and Tompa, M. (2001). Finding motifs using random projections. In *RECOMB'01 : Proceedings of the Fifth Annual International Conference on Computational Molecular Biology*, pages 67–74, Montreal, Canada.

[Burge, 1997] Burge, C. (1997). *Identification of genes in human genomic DNA*. PhD thesis, Stanford University, California, USA.

[Burge, 1998] Burge, C. (1998). Modeling dependenciesin pre-mRNA splicing signals. In Salzberg, S., Searls, D., and S.Kasif, editors, *Computational Methods in Molecular Biology*, pages 127–163. Elsevier Science.

[Burge and Karlin, 1997] Burge, C. and Karlin, S. (1997). Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.*, 268 :78–94.

[Califano, 2000] Califano, A. (2000). Splash : structural pattern localization analysis by sequential histograms. *Bioinformatics*, 16 :341–357.

[Cardon and Stormo, 1992] Cardon, L. and Stormo, G. (1992). Expectation Maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *J. Mol. Biol.*, 223 :139–170.

[Chen et al., 1995] Chen, Q., Hertz, G., and Stormo, G. (1995). MATRIX SEARCH 1.0 : a computer program that scans DNA sequences for transciptional elements using a database of weight matrices. *Comput. Appl. Biosci.*, 11 :563–566.

[Chen et al., 1998] Chen, Q., Hertz, G., and Stormo, G. (1998). *Biological sequence analysis. Probabilistic models of proteins and nucleic acids.* Cambridge University Press.

[Clift et al., 1986] Clift, B., Haussler, D., McConnell, R., Schneider, T. D., and Stormo, G. D. (1986). Sequence landscapes. *Nucleic Acids Res.*, 14 :141–158.

[Cobbs, 1995] Cobbs, A. (1995). Fast identification of approximately matching substrings. In Galil, Z. and Ukkonen, E., editors, *Combinatorial Pattern Matching*, volume 937 of *Lecture Notes in Computer Science*, pages 41–54. Springer Verlag.

[Combrugghe et al., 1984] Combrugghe, B., Busby, S., and Buc, H. (1984). Cyclic AMP receptor protein role in transcription activation. *Science*, 224 :831–838.

[Coward, 1999] Coward, E. (1999). Shufflet : Shuffling sequences while conserving the k-let counts. *Bioinformatics*, 15 :1058–1059.

[Crochemore, 1981] Crochemore, M. (1981). An optimal algorithm for computing the repetitions in a word. *Inf. Proc. Letters*, 12 :244–250.

[Crochemore et al., 2001] Crochemore, M., Hancart, C., and Lecroq, T. (2001). *Algorithmique du texte*. Vuibert.

[Crochemore and Rytter, 1994] Crochemore, M. and Rytter, W. (1994). *Text Algorithms*. Oxford University Press.

[Crochemore and Vérin, 1997] Crochemore, M. and Vérin, R. (1997). Direct construction of compact acyclic word graphs. In *Proc. of the Annual Symposium on Combinatorial Patterm Matching (CPM'97)*, Lecture Notes in Computes Sciences, pages 30–42.

[Dayhoff, 1978] Dayhoff, M. (1978). *Atlas of protein sequence and structure*, volume 5. National Biomedical Research Foundation.

[Demeler and Zhou, 1991] Demeler, B. and Zhou, G. (1991). Neural network optimization for *e.coli* promoter prediction. *Nucleic Acid Res.*, 19 :1593–1599.

[Duret and Bucher, 1997] Duret, L. and Bucher, P. (1997). Searching for regulatory elements in human noncodong sequences. *Curr. Opin. Struct. Biol.*, 7 :399–406.

[El Mabrouk, 1996] El Mabrouk, N. (1996). *Recherche approchée de motifs. Application à des séquences biologiques structurées.* PhD thesis, IGM, Univ. Marne-la-Vallée, Paris, France.

[Eskin and Pevzner, 2001] Eskin, E. and Pevzner, P. (2001). Finding composite regulatory patterns in DNA sequences. Communication personnelle, en cours de soumission.

[Ferragina and Grossi, 1995] Ferragina, P. and Grossi, R. (1995). A fully-dynamic data structure for external substring search. In *Twenty-seventh ACM Symposium on Theory of Computing (STOC'95)*, pages 693–702, Las Vegas, U.S.A. ACM Press.

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

[Fraenkel et al., 1995] Fraenkel, Y., Mandel, Y., Friedberg, D., and Margalit, H. (1995). Identification of common motifs in unaligned DNA sequences. application to *escherichia coli* Irp regulon. *Comput. Appl. Biosci.*, 11 :379–387.

[Frech et al., 1993a] Frech, K., Herrmann, G., and Werner, T. (1993a). Computer-assisted prediction classification and delimitation of protein binding sites in nucleic acids. *Nucleic Acid Res.*, 21 :1655–1664.

[Frech et al., 1993b] Frech, K., Quandt, K., and Werner, T. (1993b). Software for the analysis of DNA sequence elements of transcription. *Comput. Appl. Biosci.*, 13 :89–97.

[Galas et al., 1985] Galas, D. J., Eggert, M., and Waterman, M. S. (1985). Rigorous pattern-recognition methods for DNA sequences. analysis of promoter sequences from *escherichia coli*. *J. Mol. Biol.*, 186 :117–128.

[Garvie and Wolberger, 2001] Garvie, C. and Wolberger, C. (2001). Recognition of specific DNA sequences. *Molecular Cell*, 8 :937–946.

[Gelfand et al., 2000] Gelfand, M., Koonin, E., and Mironov, A. (2000). Prediction of transcription regulatory sites in Archae by a comparative genomic approach. *Nucleic Acid Res.*, 28 :695–705.

[Giegerich et al., 1999] Giegerich, R., Kurtz, S., and Stoye, J. (1999). Efficient implementation of lazy suffix trees. In *Proc. of the 3rd Workshop on Algorithm Engineering*, volume 1264 of *Lecture Notes in Computes Sciences*, pages 116–119, London, UK.

[Gonnet et al., 1992] Gonnet, G. H., Cohen, M. A., and Benner, S. A. (1992). Exhaustive matching of the entire protein sequence database. *Science*, 256 :1443–1445.

[Grossi and Vitter, 2000] Grossi, R. and Vitter, J. S. (2000). Compressed suffix arrays and suffix trees with applications to text indexing and string matching. In *ACM symposium on the theory of computing (STOC 2000)*, Portland, U.S.A. ACM Press.

[Guha-Thakurta and Stormo, 2001] Guha-Thakurta, D. and Stormo, G. (2001). Identifying target sites for cooperatively binding factors. *Bioinformatics*, 17 :608–621.

[Gusfield, 1997] Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology.* Cambridge University Press.

[Harley and Reynolds, 1987] Harley, C. B. and Reynolds, R. P. (1987). Analysis of *e. coli* promoter sequences. *Nucleic Acids Res.*, 15 :2343–2361.

[Helmann, 1995] Helmann, J. (1995). Compilation and analysis of *bacillus subtilis* a-dependant promoter sequences : evidence for extended contact between RNA polymerase and upstream promoter DNA. *Nucleic Acide Res.*, 23 :2351–2360.

[Henikoff and Henikoff, 1992] Henikoff, S. and Henikoff, J. (1992). Amino acids substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89 :100915–100919.

[Hertz and Stormo, 1994] Hertz, G. and Stormo, G. (1994). Identification of consensus patterns in unaligned DNA and protein sequences : A large-deviation statistical basis for penalizing gaps. In Lim, H. and Cantor, C., editors, *The third International Conference on Bioinformatics and Genome Research*, pages 201–216. World Scientific Publishing.

[Hertz and Stormo, 1996] Hertz, G. and Stormo, G. (1996). *Escherichia coli* promoter sequences : Analysis and prediction. *Meth. Enzymol.*, 273 :30–42.

[Hertz and Stormo, 1999] Hertz, G. and Stormo, G. (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15 :563–577.

[Hollon, 2001] Hollon, T. (2001). Human genes : how many ? *The Scientist*, 15(20).

[Horton and Kanehisa, 1987] Horton, P. B. and Kanehisa, M. (1987). An assessment of neural network and statistical approaches for prediction of *e.coli* promoter sites. *Nucleic Acids Res.*, 20 :4331–4338.

[Hu et al., 2000] Hu, Y.-J., Sandmeyer, S., McLaughlin, C., and Kibler, D. (2000). Combinatorial motif analysis and hypothesis generation on a genomic scale. *Bioinformatics*, 16 :222–232.

[Hui, 1992] Hui, L. C. K. (1992). Color set size problem with applications to string matching. In Apostolico, A., Crochemore, M., Galil, Z., and Manber, U., editors, *Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 230–243. Springer-Verlag.

[Hunt and Szymanski, 1977] Hunt, J. and Szymanski, T. (1977). A fast algorithm for computing longest common subsequences. *Comm. ACM*, 20 :(5)350–353.

[Jacquard, 1999] Jacquard, A. (1999). *La légende de la vie*. Flammarion.

[Jonassen, 1997] Jonassen, I. (1997). Efficient discovery of conserved patterns using a pattern graph. *Comput. Appl. Biosci.*, 13 :509–522.

[Jonassen et al., 1995] Jonassen, I., Collins, J., and Higgins, D. (1995). Finding flexible patterns in unaligned protein sequences. *Protein Science*, 4 :1587–1595.

[Karlin et al., 1989] Karlin, S., Ost, F., and Blaisdell, B. (1989). Patterns in DNA and amino-acid sequences and theur statistical significance. In Waterman, M., editor, *Mathematical Methods for DNA sequences*, pages 133–158. CRC Press.

[Klingenhoff et al., 1999] Klingenhoff, A., Frech, K., Quandt, K., and Werner, T. (1999). Functional promoter modules can be detected by formal models independant of overall nucleotides sequence similarity. *Bioinformatics 1*, 15 :180–186.

[Knuth et al., 1977] Knuth, D., Morris, J., and Pratt, V. (1977). Fast pattern matching in strings. *SIAM J. Comput.*, 6 :323–350.

[Kolchanov et al., 1998] Kolchanov, N., Ponomarenko, M., Kel, A., Kondrakhin, Y., Frolov, A., Kolpakov, F., Goryachkovsky, T., Kel, O., Ananko, E., Ignatieva, E., Podkolodnaya, O., Babenko, V., Stepanenko, I., Romashchenko, A., Merkulova, T., Vorobiev, D., Lavryushev, S., Ponomarenko, Y., Kochetov, A., Kolesov, G., Solovyev, V., Milanesi, L., Podkolodny, N., Wingender, E., and Heinemeyer, T. (1998). GeneExpress l a computer system for description, analysis, and recognition of regulatory sequences in eukaryotic genome. In *Sixth International Symposium on Intelligent Systems for Molecular Biology*, pages 95–104, Montréal, Canada. AAAI Press.

[Kondrakhin et al., 1995] Kondrakhin, Y. V., Kolchanov, N., Kel, A., Romashchenko, A., and L.Milanesi (1995). Eukaryotic promoter recognition by binding sites for transcription factors. *Comput. Appl. Biosci.*, 11 :477–488.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22 :79–86.

[Kurtz, 1999] Kurtz, S. (1999). Reducing the space requirement of suffix trees. *Software Practice and Experience*, 29 :1149–1171.

[Lafontaine and Lavery, 2001] Lafontaine, I. and Lavery, T. (2001). ADAPT : A molecular mechanics approach for studying the structural properties of long DNA sequences. *Biopolymers (Nucleic Acid Sciences)*, 56. sous presse.

[Lawrence et al., 1993] Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A., and Wooton, J. (1993). Detecting subtle sequence signals : a Gibbs sampling strategy for multiple alignement. *Science*, 262 :208–214.

[Lawrence and Reilly, 1990] Lawrence, C. E. and Reilly, A. A. (1990). An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins : struct., funct., and genetics*, 7 :41–51.

[Lefevre and Ikeda, 1994] Lefevre, C. and Ikeda, J.-E. (1994). A fast word search algorithm for the representation of sequence similarity in genomic DNA. *Nucleic Acids Res.*, 22 :404–411.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and control Theory*, 10 :707–710.

[Lewin, 2000] Lewin, B. (2000). *Genes VII - 7th edition.* Oxford University Press.

[Li et al., 1999] Li, M., Ma, B., and Wang, L. (1999). Finding similar regions in many strings. In *In proceedings of the 31st annual ACM symposium on theory of computing*, pages 473–482.

[Linhoff et al., 1997] Linhoff, M., K.L.Wright, and Ting, J.-Y. (1997). Ccaat-binding factor nf-y and rfx are required for *in vivo* assembly of a nucloprotein complex that spans 250 base pairs : the invariant chain promoter as a model. *Molecular and cellular biology*, pages 4589–4596.

[Liu et al., 2001a] Liu, R., Blackwell, T., and States, D. J. (2001a). Conformational model for binding site recognition by the *e.coli metj* transcription factor. *Bioinformatics*, 17 :622–633.

[Liu et al., 2001b] Liu, X., Brutlag, D., and Liu, J. (2001b). Bioprospector : discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. In *In proceedings of the Pacific Symposium on Biocomputings*, pages 127–138.

[Lonardi, 2001] Lonardi, S. (2001). *Global detectors of unusual words : design, implementation and applications to pattern discovery in biosequences.* PhD thesis, Purdue Univ., Indiana, USA.

[Lukashin and Borodovsky, 1998] Lukashin, A. V. and Borodovsky, M. (1998). GeneMark.hmm : new solutions for gene finding. *Nucleic Acid Res.*, 26 :1107–1115.

[Manber and Myers, 1993] Manber, U. and Myers, E. W. (1993). Suffix arrays : a new method for on-line string searches. *SIAM Journal on Computing*, 22 :935–948.

[Marsan, 1998] Marsan, L. (1998). Extraction de motifs dans les séquences d'ADN à l'aide d'un arbre des suffixes. Rapport de DEA - http ://www-igm.univ-mlv.fr/∼marsan/rapport.ps, Univ. Marne-la-Vallée, France.

[Marsan et al., 2002] Marsan, L., Pisanti, N., and Sagot, M.-F. (2002). Smile : an exact tool for extraction and evaluation of structured motif. à soumettre à Genome Research.

[Marsan and Sagot, 2001] Marsan, L. and Sagot, M.-F. (2001). Algorithms for extracting structured motifs using a suffix tree with application to promoter and regulatory site consensus identification. *J. of Comput. Biol.*, 7 :345–360.

[Masternak et al., 2000] Masternak, K., Muhlethaler-Mottet, A., Villard, J., and Zufferey, M. (2000). Ciita is a transcriptional coactivator that is recruited to mhc class ii promoters by multiple synergistic interactions with an enhanceosome complex. *Genes and development*, 14 :1156–1166.

[McCreight, 1976] McCreight, E. M. (1976). A space-economical suffix tree construction algorithm. *J. ACM*, 23 :262–272.

[Mehldau and Myers, 1993] Mehldau, G. and Myers, E. (1993). A system for pattern matching applications in biosequences. *Comput. Applic. Biosciences*, 9 :299–314.

[Moss, 2001] Moss, G. (2001). http ://www.chem.qmw.ac.uk/iupac/misc/naseq.html.

[Mulder et al., 1997] Mulder, M. A., Zappe, H., and Steyn, L. (1997). Mycobacterial promoters. *Tubercle and Lung Diseases*, 78 :(5,6) :211–223.

[Mulligan et al., 1984] Mulligan, D., Hawkley, D., Entriken, R., and McClure, W. (1984). *Escherichia Coli* promoter sequences predict in vitro RNA polymerase selectivity. *Nucleic Acid Res.*, 12 :789–800.

[Myers, 1986] Myers, E. W. (1986). An $\mathcal{O}(nd)$ difference algorithm and its variations. *Algorithmica*, 1 :251–266.

[Myers, 1994] Myers, E. W. (1994). A sublinear algorithm for approximate keyword searching. *Algorithmica*, 12 :345–374.

[Myers and Miller, 1989] Myers, E. W. and Miller, W. (1989). Approximate matching of regular expressions. *Bull. of Mathematical Biology*, 51 :5–37.

[Needleman and Wunsch, 1970] Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48 :443–453.

[Neuwald and Green, 1994] Neuwald, A. and Green, P. (1994). Detecting pattern in protein sequences. *J. Mol. Biol.*, 239 :698–712.

[Ohler et al., 1999] Ohler, U., Harbeck, S., Niemann, H., Noth, E., and Reese, M. (1999). Interpolated markov chains for eukaryotic promoter recognition. *Bioinformatics*, 15 :362–369.

[Ohler and Niemann, 2001] Ohler, U. and Niemann, H. (2001). Identification and analysis of eukaryotic promoters : recent computational approaches. *Trends in Genomic*, 17 :56–60.

[Ohmori et al., 1997] Ohmori, Y., Schreiber, R., and Hamilton, T. (1997). Synergy between interferon-γ and tumor necrosis factor α in transcriptional activation is mediated by cooperation between signal transducer and activator of transcription 1 and nuclear factor κ b. *The journal of biological chemistry*, pages 14899–14907.

[O'Neill, 1989a] O'Neill, M. (1989a). Consensus methods for finding and ranking DNA binding sites. application to *escherichia coli*. *J. Mol. Biol.*, 207 :301–310.

[O'Neill, 1989b] O'Neill, M. (1989b). *Escherichia Coli* promoters. I. consensus as it relates to spacing class specificity repeat substructure and three-dimensional organization. *J. Biol. Chem.*, 264 :5522–5530.

[O'Neill and Chiafari, 1989] O'Neill, M. and Chiafari, F. (1989). *Escherichia Coli* promoters. II. a spacing-class dependant promoter search protocol. *J. Biol. Chem.*, 264 :5531–5534.

[O'Neill and Chiafari, 1991] O'Neill, M. and Chiafari, F. (1991). Training back-propagation neural network to define and detex DNA-binding sites. *Nucleic Acid Res.*, 19 :313–318.

[Ozoline et al., 1998] Ozoline, O., Deev, A. A., and Arkhipova, M. (1998). Non-canonical sequence elements in the promoter structure. cluster analysis of promoters recognized by *escherichia coli* in RNA polymerase. *Nucleic Acid Res.*, 25 :4703–4709.

[Parida et al., 2000] Parida, L., Rigoutsos, I., Floratos, A., Platt, D., and Gao, Y. (2000). Pattern discovery on character sets and real-valued data : linear bound on irredundant motifs and plynomial time algorithms. In *In proceedings of the eleventh ACM-SIAM symposium on discrete algorithms (SODA 2000)*, pages 297–308. ACM Press.

[Pavesi et al., 2001] Pavesi, G., Mauri, G., and Pesole, G. (2001). An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, 17 :S207–S214.

[Pearson, 1990] Pearson, W. (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. *Meth. Enzymol.*, 183 :63–98.

[Pearson and Lipman, 1988] Pearson, W. and Lipman, D. (1988). Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85 :2444–2448.

[Perez-Rueda et al., 1998] Perez-Rueda, E., Gralla, J., and Collado-Vides, J. (1998). Genomic position analysis and the transcription machinery. *J. Mol. Biol.*, 275 :165–170.

[Pevzner and Sze, 2000] Pevzner, P. and Sze, S. (2000). Combinatorial approaches to finding subtle signals in DNA sequences. In *In proceedings of the eight conference on Intelligent Systems for Molecular Biology*, pages 269–278, San Diego, CA, USA. AAAI Press.

[Prestridge, 1993] Prestridge, D. (1993). Predicting Pol II promoter sequence using transcription factor binding sites. *J. Mol. Biol.*, 249 :923–932.

[Prestridge and Stormo, 1991] Prestridge, D. and Stormo, G. (1991). SIGNAL SCAN : a computer program that scans DNA sequences for eukaryotic transcriptional elements. *Comput. Appl. Biosci.*, 7 :203–206.

[Prestridge and Stormo, 1993] Prestridge, D. and Stormo, G. (1993). SIGNAL SCAN 3.0 : new database and program features. *Comput. Appl. Biosci.*, 9 :113–115.

[Quandt et al., 1996] Quandt, K., Grote, K., and Werner, T. (1996). GenomeInspector : basic software tools for analysis of spatial correlations between genomic structures within megabase sequences. *Genomics*, 33 :301–304.

[Queen et al., 1982] Queen, C., Wegman, M., and Korn, L. (1982). Improvements to a program for DNA analysis. a procedure to find homologies among many sequences. *Nucleic Acid Res.*, 10 :449–456.

[Record et al., 1996] Record, M., Reznikoff, W., Craig, M., McQuade, K., and Schlax, P. (1996). *Escherichia coli* RNA polymerase σ^{70} promoters, and the kinetics of the steps of transcription initiation. In Neidhardt, F., editor, *Escherichia coli and Salmonella, volume 1*, pages 792–820. ASM Press.

[Régnier and Szpankowski, 1997] Régnier, M. and Szpankowski, W. (1997). On the approximate pattern occurrences in a text. In Society, I. C., editor, *Compression and Complexity of SEQUENCES 1997*, pages 253–264. In Proceedings SEQUENCE'97,Positano, Italy.

[Reinert and Schbath, 1998] Reinert, G. and Schbath, S. (1998). Compound poisson and poisson process approximations for occurrences of multiple words in markov chains. *J. Comp. Biol.*, 5 :223–253.

[Reinert et al., 2000] Reinert, G., Schbath, S., and Waterman, M. (2000). Probabilistic and statistical properties of words. *J. Comp. Biol.*, 7 :1–46.

[Robin et al., 2001] Robin, S., Daudin, J.-J., Richard, H., Sagot, M.-F., and Schbath, S. (2001). Occurrence probability of structured motifs in random sequences. En cours de soumission.

[Rocke, 2000] Rocke, E. (2000). Using suffix trees for gapped motif discovery. In Giancarlo, R. and Sankoff, D., editors, *Lecture Notes in Computer Science*, volume 1848. Springer-Verlag.

[Rocke and Tompa, 1998] Rocke, E. and Tompa, M. (1998). An algorithm for finding novel gapped motifs in DNA sequences. In Istrail, S., Pevzner, P., and Waterman, M., editors, *RECOMB'98. Proceedings of Second Annual International Conference on Computational Molecular Biology*, pages 228–233. ACM Press.

[Roth et al., 1998] Roth, F. P., Hughes, J. D., Estep, P., and Church, G. (1998). Finding dna regulatory motifs within unaligned noncoding sequences clustered by whole-genome mrna quantitation. *Nat. Biotechnol.*, 16 :939–945.

[Sacks, 1988] Sacks, O. (1988). *L'homme qui prenait sa femme pour un chapeau.* Seuil.

[Sagot, 1993] Sagot, M.-F. (1993). Ressemblance lexicale et structurale entre macromolécules - formalisation et approches combinatoires. Mémoire de DEA d'Informatique Fondamentale - Université de Paris VII, France.

[Sagot, 1996] Sagot, M.-F. (1996). *Ressemblance lexicale et structurale entre macromolécules - Formalisation et approches combinatoires.* PhD thesis, IGM, Univ. Marne-la-Vallée, Paris, France.

[Sagot, 1998] Sagot, M.-F. (1998). Spelling approximate repeated or common motifs using a suffix tree. In Lucchesi, C. L. and Moura, A. V., editors, *LATIN9́8 : Theoretical Informatics*, Lecture Notes in Computer Science, pages 111–127. Springer-Verlag.

[Sagot and A.Viari, 1997] Sagot, M.-F. and A.Viari (1997). Flexible identification of structural objects in nucleic acid sequences : palindromes, mirror repeats, pseudoknots and triple helices. In Hein, J., editor, *Comb. Pattern Matching*, Lecture Notes in Computer Science, pages 224–246. Springer-Verlag.

[Sagot et al., 1995a] Sagot, M.-F., Escalier, V., Viari, A., and Soldano, H. (1995a). Searching for repeated words in a text allowing for mismatches and gaps. In Baeza-Yates, R. and Manber, U., editors, *Second South American Workshop on String Processing*, pages 87–100, Viñas del Mar, Chili. University of Chili.

[Sagot and Myers, 1998] Sagot, M.-F. and Myers, E. W. (1998). Identifying satellites in nucleic acid sequences. *J. of Comp. Biol.*, 5 :(3)539–554.

[Sagot and Viari, 1996] Sagot, M.-F. and Viari, A. (1996). A double combinatorial approach to discovering patterns in biological sequences. In Hirschberg, D. and Myers, G., editors, *Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 186–208. Springer-Verlag.

[Sagot et al., 1995b] Sagot, M.-F., Viari, A., Pothier, J., Escalier, V., and Soldano, H. (1995b). Multiple comparison in biology : some mathematical formalizations of the problem and combinatorial approaches to solve it. Rapport Interne 95-7, Atelier de BioInformatique, Université de Paris VI.

[Sagot et al., 1995c] Sagot, M.-F., Viari, A., and Soldano, H. (1995c). A distance-based block searching algorithm. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T., and Wodak, S., editors, *Third International Symposium on Intelligent Systems for Molecular Biology*, pages 322–331, Cambridge, England. AAAI Press.

[Sagot et al., 1997] Sagot, M.-F., Viari, A., and Soldano, H. (1997). Multiple comparison : a peptide matching approach. *Theoret. Comput. Sci.*, 180 :115–137. presented at *Combinatorial Pattern Matching 1995.*

[Sagot and Wakabayashi, 2001] Sagot, M.-F. and Wakabayashi, Y. (2001). Pattern inference under many guises. In Reed, B. and Sales, C. L., editors, *Fortaleza Summer School on Algorithms and Combinatorics*, Fortaleza, Brésil. À paraître dans Lecture Notes in Computer Science, Springer-Verlag, 2001.

[Salzberg et al., 1999] Salzberg, S. L., Pertea, M., Delcher, A. L., Gardner, M. J., and Tettelin, H. (1999). Interpolated markov models for eukaryotic gene finding. *Comput. Appl. Biosci.*, 59 :24–31.

[Saurin and Marlière, 1987] Saurin, W. and Marlière, P. (1987). Matching relational patterns in nucleic acid sequences. *Comput. Appl. Biosci.*, 3 :115–120.

[Schneider and Stormo, 1986] Schneider, T. and Stormo, G. (1986). Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188 :415–431.

[Sewell and Durbin, 1995] Sewell, R. and Durbin, R. (1995). Method for calculation of matching a bounded regular expression in a random data string. *J. Comput. Biol.*, 2 :25–34.

[Sinha, 2001] Sinha, S. (2001). Composite motifs in promoter regions of genes : models and algorithms. Technical report, Dep. of computer science and engineering, Univ. of Washington.

[Sinha and Tompa, 2000] Sinha, S. and Tompa, M. (2000). A statistical method for finding transcription factor binding sites. In *Eight International Conference on Intelligent Systems for Molecular Biology*, pages 344–354, San Diego, CA, USA. AAAI Press.

[Staden, 1984] Staden, R. (1984). Computer methods to locate signals in nucleic acid sequences. *Nucleic Acid. Res.*, 12 :505–519.

[Staden, 1989] Staden, R. (1989). Methods for discovering novel motifs in nucleic acid sequences. *Comput. Appl. Biosci.*, 5 :293–298.

[Stormo, 1988] Stormo, G. D. (1988). Computer methods for analyzing sequence recognition of nucleic acids. *Annu. Rev. Biophys. Biophys. Chem*, 17 :241–263.

[Stormo, 1990] Stormo, G. D. (1990). Consensus pattern in DNA. *Meth. Enzymol.*, 183 :211–221.

[Stormo and Hartzell III, 1989] Stormo, G. D. and Hartzell III, G. W. (1989). Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. USA*, 86 :1183–1187.

[Sze et al., 2002] Sze, S.-H., Gelfand, M., and Pevzner, P. (2002). Finding weak motifs in DNA sequences. In *Pacific Symposium on Biocomputing (PSB'2002).* À paraître.

[Tagle et al., 1988] Tagle, D., Koop, B., Goodman, M., Slightom, J., and Jones, R. (1988). Embryonic e and g globin genes of a prosimian primate (galago crassicaudatus) : Nucleotide and amino acid sequences, developmental regulation and phylogenetic footprints. *J. Mol. Biol.*, 203 :439–455.

[Thijs et al., 2001a] Thijs, G., Lescot, M., Marchal, K., Rombauts, S., Moor, B. D., Rouzé, P., and Moreau, Y. (2001a). A higher order background model improves the detection of regulatory elements by gibbs sampling. *Bioinformatics, in press.*

[Thijs et al., 2001b] Thijs, G., Marchal, K., Lescot, M., Rombauts, S., Moor, B. D., Rouzé, P., and Moreau, Y. (2001b). A gibbs sampling method to detect over-represented motifs in upstream regions of coexpressed genes. In *Proceedings RECOMB'2001.*

[Tompa, 1999] Tompa, M. (1999). An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In *Seventh Intl. Conf. on Intelligent Systems for Molecular Biology*, pages 262–271, Heidelberg, Germany. AAAI Press.

[Tompa, 2001] Tompa, M. (2001). Identifying functional elements by comparative DNA sequence analysis. *Genome Research*, 11 :1143–1144.

[Ukkonen, 1985] Ukkonen, E. (1985). Algorithm for approximate string matching. *Inform. and Control*, 64 :100–118.

[Ukkonen, 1993] Ukkonen, E. (1993). Approximate string matching over suffix trees. In A. Apostolico, M. Crochemore, Z. G. and Manber, U., editors, *Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 228–242. Springer-Verlag.

[Ukkonen, 1995] Ukkonen, E. (1995). On-line construction of suffix-trees. *Algorithmica*, 14 :(3)484–492.

[Ulyanov and Stormo, 1995] Ulyanov, A. and Stormo, G. (1995). Multi-alphabet consensus algorithm for identification of low specificity protein-DNA interactions. *Nucleic Acid Res.*, 23 :1434–1440.

[van Helden et al., 1998] van Helden, J., Andre, B., and Collado-Vides, J. (1998). Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotides frequencies. *J. Mol. Biol.*, 281 :827–842.

[van Helden et al., 2000] van Helden, J., Rios, A., and Collado-Vides, J. (2000). Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acid Res.*, 28 :827–842.

[Vanet et al., 2000] Vanet, A., Marsan, L., Labigne, A., and Sagot, M.-F. (2000). Infering regulatory elements from a whole genome. an application to the analysis of genome of *helicobacter pylori* σ_{80} family of promoter signals. *J. Mol. Biol.*, 297 :335–353.

[Vanet et al., 1999] Vanet, A., Marsan, L., and Sagot, M.-F. (1999). Promoter sequences and algorithmical methods for identifying them. *Research in Microbiology*, 150 :779–799.

[Waterman, 1984] Waterman, M. S. (1984). General methods of sequence comparison. *Bull. Math. Biol.*, 46 :473–500.

[Waterman, 1986] Waterman, M. S. (1986). Multiple sequence alignments by consensus. *Nucleic Acids Res.*, 14 :9095–9102.

[Waterman, 1989] Waterman, M. S. (1989). Consensus patterns in sequences. In Waterman, M. S., editor, *Mathematical Methods for DNA Sequences*, pages 93–116. CRC Press.

[Waterman et al., 1984] Waterman, M. S., Arratia, R., and Galas, D. J. (1984). Pattern recognition in several sequences : consensus and alignment. *Bull. Math. Biol.*, 46 :515–527.

[Werner, 1999] Werner, T. (1999). Models for prediction and recognition of eukaryotic promoters. *Mammalian Genome*, 10 :168–175.

[Wolferstetter et al., 1996] Wolferstetter, F., Frech, K., Herrmann, G., and Werner, T. (1996). Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithm. *Comp. Appl. Biosci.*, 12 :71–80.

[Wu and Manber, 1992] Wu, S. and Manber, U. (1992). Fast text searching allowing errors. *Commun. ACM*, 35 :83–91.

[Yada et al., 1998] Yada, T., Totoki, Y., Ishikawa, M., Asai, K., and Nakai, K. (1998). Automatic extraction of motifs represented in the hidden markov model from a number of DNA sequences. *Bioinformatics*, 14 :317–325.

Index