THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE MARNE-LA-VALLÉE

Spécialité : Informatique Fondamentale

présentée par

Marie-France SAGOT

pour l'obtention du titre de DOCTEUR DE L'UNIVERSITÉ DE MARNE-LA-VALLÉE

Ressemblance lexicale et structurale entre macromolécules - Formalisation et approches combinatoires

Soutenue le 9 Juillet 1996 devant le jury composé de :

Pr. Dominique Perrin (Président)
Pr. Didier Arquès (Rapporteur)
Dr. Antoine Danchin (Rapporteur)
Pr. Eugene W. Myers (Rapporteur)
Pr. David Sankoff (Rapporteur)
Pr. Maxime Crochemore (Examinateur)
Dr. Alain Viari (Examinateur)
Pr. Ricardo Baeza-Yates (Invité)

Ce travail est dédié

à mon père qui nous manque tellement

à ma mère et à mon frère

aux amis brésiliens, français et d'ailleurs

et à Alain Viari

Our share of night to bear -
Our share of morning -
Our blank in bliss to fill
Our blank in scorning -

Here a star, and there a star,
Some lose their way!
Here a mist, and there a mist,
Afterwards - Day!

Poème #113. Emily Dickinson
The Complete Poems of Emily Dickinson
Thomas H. Johnson (Ed.). Faber and Faber, 1975

Remerciements

je tiens à remercier le Professeur Dominique Perrin de m'avoir si gentiment accueillie dans le cadre du DEA d'Informatique Fondamentale de Paris 7 et de m'avoir fait la très grande joie d'accepter de présider le jury de cette thèse

je remercie très chaleureusement le Professeur Maxime Crochemore pour avoir accepté de diriger ce travail depuis l'époque du DEA, pour sa disponibilité et l'aide si précieuse qu'il m'a apportée lors de la rédaction des articles qui en sont issus. J'ai été particulièrement heureuse de l'opportunité fournie par un congrès en Amérique Latine de mieux le connaître et de lui faire connaître un peu des gens de mon pays

j'adresse tous mes remerciements à Antoine Danchin pour avoir accepté d'être rapporteur de cette thèse ainsi que pour tous les jeudis matins où il a su me donner tellement envie de continuer sur cette voie

je remercie très vivement les Professeurs Eugene W. Myers et David Sankoff du grand honneur qu'ils m'ont fait d'accepter de participer au jury et d'en être les rapporteurs

je remercie sincèrement le Professeur Didier Arquès de s'être intéressé à ce travail et avoir bien voulu en être rapporteur

je tiens à remercier tout particulièrement Alain Viari pour son intelligence de tous les domaines, son exigence et son indépendance d'esprit, pour être quelqu'un capable également d'une grande attention et délicatesse, une personne extraordinaire dont la rencontre et la confiance m'ont très profondément marquée

je remercie le Professeur Ricardo Baeza-Yates pour avoir accepté d'assister à la soutenance et pour sa gentillesse et chaleur humaine les deux fois où j'ai eu l'occasion de le rencontrer

je remercie très chaleureusement Henri Soldano pour son soutien matériel et psychologique si importants, et pour l'intérêt très vif qu'il a toujours porté à ce travail : celui-ci a énormément bénéficié de ses idées et conseils

je remercie très fortement Claude Thermes et Yves d'Aubenton Carafa pour leur gentillesse, cela a été un vif plaisir de travailler eux et j'espère que nous pourrons continuer à le faire pendant longtemps encore

je remercie Madame Monique Pagès sans qui l'Atelier de BioInformatique n'aurait jamais existé

je remercie Abdel Laoui, Evelyne James Surcouf et Jean-Loup Risler pour le soutien financier apporté par Organibio

je remercie toutes les autres personnes directement attachées ou très proches de l'Atelier de BioInformatique et qui m'ont beaucoup soutenues tout au long de cette thèse :

> Anne-Marie Leseney dont la gaieté et le grand coeur font de ses visites au laboratoire une source de joie

Claudine Médigue pour sa gentillesse et les conseils qu'elle m'a donnés, et pour le sourire qui vous remonte toujours le moral

Corinne Millerat pour sa simplicité, son amitié si chaleureuse et son constant soutien qui m'ont été tellement importants, en particulier dans les derniers mois de ce travail - je lui dois énormément

Marie-Françoise Lancastre pour son amitié, son franc parler et pour nos nombreuses conversations qui m'ont été très précieuses

Pascale Jean pour sa jeunesse, sa spontanéité et pour apporter au laboratoire une ambiance très tonique et vivifiante

Bernard Billoud pour tout ce qu'il m'a appris en biologie, pour sa constante gentillesse, son attention, son humour et sa sérénité

Dominique Bouthinon pour sa disponibilité et pour les encouragements qu'il m'a prodigués tout au long de cette thèse

Joël Pothier pour sa constante bonne humeur et disponibilité et pour l'appui solide qu'il a toujours représenté, sa seule présence au laboratoire a souvent été une source de grand réconfort

Stéphane Tsacas pour m'avoir très gentiment laissé utiliser 'snaut', n'avoir jamais rien dit quand j'y 'dumpait' des 'cores' énormes (les termes en anglais ont un effet tellement plus frappant) et enfin pour être toujours venu me prévenir lorsqu'il tuait Alice

Vincent Escalier pour sa gentillesse, pour son esprit très vif et son constant soutien : cela a été un réel plaisir de l'avoir eu pour compagnon de thèse

ceux qui n'y viennent que la nuit, Christiane et Peggy, ou qui sont toujours de passage, F.R., Pierre et Marc

je tiens à remercier également les membres de l'Institut Gaspard Monge de m'y avoir toujours accueilli avec beaucoup de gentillesse, tout particulièrement :

Frédérique Bassino pour sa bonne humeur permanente et son amitié depuis l'époque du DEA

Giuseppina Rindone pour son attention et l'appui très solide qu'elle a représenté dans les derniers mois de la thèse

Nadia El Mabrouk pour sa très grande gentillesse, sa simplicité et son amitié - j'espère vivement que nous pourrons venir un jour à travailler ensemble

Olivier Carton avec qui c'est toujours une grande joie de partager les repas (mais il vaut mieux apporter plus de chocolat) et qui me donne toujours de très bons conseils (travailler moins, etc)

mes co-responsables des TDs et TPs de M3MI, et tout particulièrement Chan, pour le soutien très concret qu'ils m'ont apporté alors que j'abordais les derniers mois de cette thèse

et, tout spécialement, Line Fonfrède pour son immense gentillesse et sa constante disponibilité qui effacent toutes les angoisses

je remercie très chaleureusement Emmanuelle Ollivier et Marie-Odile Delorme pour leur soutien, amitié et générosité - ce serait une très grande joie de pouvoir travailler avec elles et d'être plus souvent ensembles

je remercie très sincèrement Alain Guénoche et Jacques Haiech pour l'intérêt qu'ils ont toujours porté à ce travail et pour leur gentillesse (et Jacques pour une conversation qui m'a beaucoup aidée à l'époque même si ses pronostics se sont révélés être trop optimistes)

je remercie Pierre Nicodème pour ses encouragements, l'intérêt qu'il porte à mon travail et pour son amitié

je remercie Frédérique Lisacek pour sa spontanéité et sa joie de vivre si contagieuse

je remercie Chrystelle Sanlaville et Véronique Bruyère pour leur très grande gentillesse et la façon simple et directe avec laquelle elles m'ont offert leur amitié

je remercie les organisateurs de LATIN'95 de l'opportunité qu'ils m'ont fournie de revoir l'Amérique Latine ainsi que certaines personnes de mon université d'origine et pays - cela a représenté un des moments les plus heureux et importants de ces deux ans et demi de thèse

je remercie les personnes souvent rencontrées pendant la durée de cette thèse, à l'ABI ou lors de réunions de travail et de congrès : Christine Gaspin, Claudine Devauchelle, Fariza Tahi, Fouzia Moussouni, Gwennaele Fichant, Véronique Biaudet, Eric Rivals, Ivan Mosser, Jacques Lebbe, Joël Quinqueton, Karel Zimmermann, Laurent Duret, Laurent Mouchard, Pierre Rouze, Thierry Vermat, Vincent Ferretti et Yves Quentin

je remercie les Professeurs de l'Instituto de Matemática e Estatística de l'Université de São Paulo, Brésil, pour leur enthousiasme et soutien toutes ces années malgré la distance, tout particulièrement les Professeurs Arnaldo Mandel, Imre Simon, Ivan de Camargo, José Coelho de Pina Jr., Kunio Okuda, Nami Kobayashi, Paulo Feofiloff, Siang Wun Song, Valdemar W. Setzer et Yoshiko Wakabayashi, ainsi que João Meidanis de l'Université de Campinas. J'espère très vivement que nous pourrons un jour établir de solides collaborations et que je réussirais ainsi à rapprocher un peu les deux mondes auxquels j'appartiens et qui me tiennent tant à coeur.

je remercie tout spécialement les amis du Brésil, ou de l'"époque brésilienne', qui, malgré une distance si grande, ont toujours fait preuve d'une totale fidélité, m'ont constamment soutenue et encouragée et, à un moment particulièrement difficile, littéralement 'portée' :

Alexandre, Eduardo et Sílvia

Claudia, Denise, Diana, Eliane, Eugenio, Flávio, França, Geni, Humberto, Kelly, Luciana (Yurica Omae), Marcia, Marcos Jorge, Maurício, Orlando, Renata, Rogério, Soon, Ubiratan et Vivian

Júlio, Reinaldo et Ricardo

Jan, Jennie, Joyce et Pia

Heraldo à qui je dois beaucoup

Luciana Camargo de Andrade pour son esprit libre

je remercie très fortement Cyro de Carvalho Patarra pour son amitié constante - il me manque beaucoup

enfin je remercie tout spécialement ma famille, mon frère, mes tantes brésiliennes et française et ma mère pour avoir toujours été là et m'avoir donné des valeurs solides

et, surtout,

je remercie mon père qui avec infiniment de patience, compréhension et tendresse m'a aidé à sortir de l'isolement. Ce travail lui appartient.

Table des matières

Chapitre 1

Introduction

"Would you tell me, please, which way I ought to go from here?"
"That depends a great deal on where you want to go," said the Cat.
"I don't much care where -" said Alice.
"Then it doesn't matter which way you go," said the Cat.
"- so long as I get somewhere," Alice added as an explanation.
"Oh, you're sure to do that," said the Cat, "if you only walk long enough."

Alice in Wonderland. Lewis Carroll.

Imaginez que l'on place devant vous une image — par exemple, la photographie d'une scène de la vie quotidienne un jour de printemps autour du bassin du Jardin du Luxembourg — et que l'on vous demande de dire si cette image contient un petit chien, et si oui, où se situe l'animal. Supposez que l'on vous fournisse un dessin du chien afin de vous guider, et que ce dessin soit identique à celui du petit chien illustré dans l'image (s'il s'y trouve). Dans ce cas, il suffit de glisser un décalque de ce dessin sur toute l'étendue de l'image pour être à même de répondre à la question que l'on vous a posée. Si, par contre, le chien du dessin que l'on vous a donné apparaît de façon différente de celui que vous devez essayer de localiser (par exemple, debout de face, alors que dans l'image le chien est couché), il vous faudra repérer d'abord quelles sont les caractéristiques essentielles de l'animal (sa forme, son apparence générale) afin que le dessin puisse vous être utile dans votre examen de l'image. Enfin, si rien ne vous est fourni comme aide dans votre recherche, il vous faudra recréer un dessin mental de ce qui fait d'un chien un chien (et non un chat ou une souris) et c'est ce 'dessin' qui doit alors vous guider. Notez que j'ai dit recréer car si vous n'avez jamais vu un chien dans votre vie, ni lu une description de l'animal, vous serez impuissant à répondre à la question, même si l'image contient vraiment un chien. Dans tous les cas, il vous faut donc avoir une idée au moins *a priori*, concrète ou abstraite, de ce que vous cherchez.

Imaginez maintenant que l'on pose devant vous non plus une image, mais toute une série de scènes — toujours de notre bassin du parc — dont chacune capte cette fois un instant différent de la vie, par exemple trente photographies, chacune représentant un jour du mois de juin à la même heure du matin. Ce que l'on vous demande à présent est de dire si vous voyez quelque chose d'inhabituel dans ces cinquantes images. La première question qui vous vient à l'esprit est sans doute, qu'est-ce que l'inhabituel? Cela peut être une singularité, quelque chose qui n'est aperçue qu'une fois. Parmi toutes ces images qui fourmillent de détails, beaucoup d'objets sont singuliers, mais, par exemple, ce premier lundi de juin, toutes les chaises étaient renversées par terre et cela attire votre attention. Si vous avez une idée des choses qui sont 'normales' dans

la vie, vous vous dites que cela est certainement inhabituel. Bien sûr, il vous faut avoir l'idée d'abord.

L'inhabituel parmi tous ces objets et personnages qui changent d'un jour à l'autre peut cependant être considéré non comme quelque chose de rare — un évènement comme celui des chaises renversées — mais plutôt comme quelque chose qui réapparaît de manière régulière. Dans toutes ces images vous observez par exemple un petit garçon qui regarde tout joyeux un voilier en papier glisser à la surface de l'eau. C'est toujours le même petit garçon, le même voilier tout blanc, la même expression de bonheur.

Identifier ce second type d"inhabituel' est plus facile qu'identifier le premier. Ce n'est pas aussi facile que de localiser un chien (dont on possède un décalque ou une simple description) parce que l'objet qui apparaît souvent n'est pas connu à l'avance. Dans ces images, c'est un petit garçon et son voilier, ailleurs ce sera autre chose. Une comparaison des images, surtout si l'on dispose de beaucoup d'images (et d'une bonne mémoire), facilite cependant la recherche. La finesse de la réponse que nous fournissons à la question : dites-moi ce qui est inhabituel parmi les images, dépend toutefois encore beaucoup de pouvoir avoir une idée *a priori* de ce qui est important parmi tous les détails que ces images présentent. Par exemple, les chaises apparaissent tous les jours, mais vous ne serez sans doute pas tenté d'avoir votre attention attirée par cela car vous savez que, dans toutes les photographies de parcs, n'importe où dans le monde, des chaises similaires seront présentes. Mais il faut que vous sachiez cela pour pouvoir tracer la frontière entre ce qui est juste inhabituellement fréquent, et ce qui est beaucoup trop fréquent et donc probablement sans importance. Parfois également, ce qui va compter n'est pas un objet qui se répète, mais plusieurs objets qui sont vus souvent ensemble. Ainsi, vous remarquez que tous les mardis et mercredis, le petit garçon apparaît avec une écharpe rouge autour du cou. Or ces jours-là justement — ces jours-là uniquement — vous observez également une vieille dame assise sur une des chaises du parc, proche de l'enfant. C'est sa grand-mère, c'est elle aussi qui explique l'écharpe : elle a trop peur qu'il prenne froid. Pour vous, c'est peut être cela qui est inhabituel, et intéressant, dans ces images : le fait qu'écharpe rouge et vieille dame sont présents toujours ensemble (association d'objets) et que la vieille dame est assise près de l'enfant (lieu de la répétition). Bien sûr cela aurait pu être une simple coïncidence : à l'heure où l'enfant joue dans le parc, se trouve présent également un vieux monsieur avec une casquette verte. Les deux ne vont pourtant pas ensemble, mais vous n'avez aucun moyen de le savoir. Ce que vous êtes capable de dire c'est uniquement : cet objet, ou personnage, apparaît tant de fois dans toutes ces images. Par contre, au contraire de ce qui se passe dans la recherche d'un objet spécifique qui doit être connu *a priori*, ici vous pouvez identifier des choses surprenantes, des choses que vous ne connaissiez pas à l'avance : il suffit qu'elles se répètent suffisamment souvent pour attirer votre regard, et que vous soyez sans parti pris. Car en effet, si vous n'aimez pas les oiseaux, vous ne dépenserez sans doute pas de temps à les regarder et vous ne remarquerez pas le petit rouge-gorge qui tous les jours vient se poser au bord du bassin.

Les 'images' qui vont être le sujet de ce travail, celles que nous allons étudier et analyser au moyen d'algorithmes, représentent des macromolécules biologiques. Celles qui vont nous intéresser sont de deux types : les acides nucléiques (ADN et ARN) et les protéines. Les premières peuvent être vues comme les agents transporteurs et transmetteurs du matériel génétique nécessaire au maintien de la vie, et les secondes comme les agents exécuteurs des instructions contenues dans ce matériel. Nous avons choisi pour l'instant de ne les regarder qu'à travers une fenêtre qui nous permet de les voir comme des chaînes de caractères sur un certain alphabet. Cette fenêtre est 'naturelle' car bien que ces macromolécules soient en fait des objets tridimensionnels, ce sont également des polymères linéaires composés d'éléments plus simples — des monomères appelés nucléotides (de quatre types différents) dans le cas de

l'ADN et de l'ARN, et des acides aminés (au nombre de vingt) dans le cas des protéines. Cette vision unidimensionnelle des macromolécules ne nous empêche pas de les voir également comme des formes dans l'espace, mais ce sont alors des formes contraintes par l'ordre de succession des monomères le long de la chaîne et codées sur un alphabet d'angles discrétisés (ceux que réalise la chaîne en se repliant sur elle-même). Dans tous les cas, ce sont les symboles associés à chacun de ces monomères (en tant que tels ou en tant qu'objets d'une chaîne 3D) qui vont composer notre alphabet. L'analyse que nous allons faire des macromolécules est donc une analyse lexicale.

La recherche de ce qui est commun à un ensemble d'objets présente une analogie avec ce dont nous avons discuté auparavant à propos de la recherche d'objets communs à diverses images d'un parc. En effet, si nos 'images' ne vont pas représenter le même objet, suggérant que nous aurions dû parler d'images de parcs différents plutôt que de celles d'un même parc à divers moments de la vie, le temps va souvent constituer un lien entre les objets qui vont nous intéresser ici. Ce temps est celui de l'évolution et les macromolécules que nous allons vouloir comparer peuvent être vues comme des 'copies', ou des copies de copies différentes d'une même macromolécule ou d'une même forme ancestrale, chaque copie étant le résultat d'une suite particulière de transformations et de mutations effectuées sur des copies antérieures. Certaines parties de ces copies cependant vont 'changer moins' que d'autres, et nous savons que ces parties sont justement celles qui ont le plus de chance d'être associées, soit à une activité de la chaîne polymérique, soit à un élément structural qui sert de charpente à la structure générale que la macromolécule adopte dans l'espace. Ce sont essentiellement ces parties qui nous intéressent et dans notre analyse lexicale des chaînes de symboles codant pour ces molécules, nous allons donc vouloir identifier des mots dans ces chaînes similaires entre eux d'une certaine façon.

Une première difficulté à laquelle nous allons être confrontés dans cette tâche est alors de définir plus précisément ce que ces termes vagues de 'changent moins' et 'similaires entre eux d'une certaine façon' veulent dire. Une partie essentielle de ce travail est ainsi composée par une exploration de la notion de ressemblance entre macromolécules biologiques. Les monomères sont le point de départ de cette exploration et leur ressemblance est essentiellement une fonction des propriétés physico-chimiques que certains d'entre eux possèdent en commun. La relation cependant n'est pas simple car le rôle exercé par ces monomères dans une macromolécule dépend de plusieurs autres facteurs que leur seule identité, comme leur position dans la chaîne et les liaisons chimiques qu'ils réussissent à établir avec d'autres monomères. Cette influence du contexte ne peut être captée pour l'instant que de manière indirecte et locale. Aussi d'une façon générale, nous devons considérer que les mesures de similarité que nous pouvons établir sont indépendantes du contexte. Ces mesures sont soit discrètes (deux monomères sont considérés comme semblables ou non), soit représentent une valeur numérique. C'est sur la base de ces mesures de similarité entre monomères que nous définissons alors celles entre mots dans les chaînes.

Deux notions sont fondamentales dans les définitions de ressemblance que nous établissons : l'une est celle d'une comparaison multiple, l'autre celle du modèle. L'idée de la première est que comparer un grand nombre d'objets simultanément nous permet d'être beaucoup plus sensibles, c'est-à-dire, de détecter de plus faibles similarités entre ces objets. Le second concept est ce qui permet de réaliser cette comparaison simultanée de manière efficace. Ce n'est pas un concept nouveau, nous n'avons fait que l'étendre et l'adapter à nos besoins, et le rendre le plus souple possible. Cet objet, qui est externe aux chaînes que l'on compare, peut être soit un mot sur le même alphabet, soit un ensemble de mots, c'est-à-dire, un produit cartésien de sous-ensembles des symboles composant l'alphabet des chaînes. La façon dont nous allons nous en servir pour notre propos consiste à dire que les mots d'un ensemble ne sont similaires

entre eux que s'il existe un modèle auquel tous ressemblent. Réciproquement, un modèle n'est considéré comme intéressant ('inhabituel') que s'il existe au moins un certain nombre de mots appartenant à des chaînes différentes qui ressemblent à ce modèle (nous disons de ces mots qu'ils sont des occurrences du modèle dans les chaînes et du modèle qu'il est présent dans les chaînes correspondantes). Notre recherche de mots similaires communs se résume alors à une recherche efficace de ces modèles présents dans au moins un certain nombre des chaînes de l'ensemble que nous comparons. Comme pour les images du parc, nous repérons les modèles individuellement. Si deux ou plus de ces modèles sont identifiés par rapport à un même ensemble de chaînes, nous ne sommes pas capables pour le moment de dire si leur association a un sens ou est fortuite.

Notre problème de fournir une définition de la ressemblance entre mots devient donc à présent celui de définir la ressemblance entre des mots et un modèle. Nous établissons en fait plusieurs de ces définitions dans ce travail, pour des raisons à la fois historiques et pragmatiques. La raison historique est que ces définitions ont, elles aussi, évoluées et se sont affinées au fil du temps. Certaines des définitions initiales, par exemple celle ayant pour base une distance d'édition entre mots, se sont retrouvées contenues dans celles établies plus tard, d'autres sont demeurées à part et répondent à des besoins différents. En effet, il n'y a sans doute pas une seule définition possible de la ressemblance, et chacune est adaptée à un problème particulier. Par pragmatisme donc, nous en maintenons plusieurs. Un souci est cependant demeuré constant par rapport à toutes : que ces définitions soient mathématiquement précises. L'idée est de donner au biologiste (la personne intéressée par ce travail) les moyens d'interpréter avec le moins d'ambiguité possible les résultats obtenus par une application de ces définitions. Pour cela, il faut également que les algorithmes qui recherchent les modèles possèdent certaines caractéristiques.

La principale de ces caractéristiques est l'exhaustivité de la recherche qu'ils effectuent. Une autre partie essentielle de ce travail a ainsi porté sur l'élaboration d'algorithmes combinatoires qui explorent tout l'espace des chaînes que l'on compare et qui fournissent en résultat une liste de tous les modèles, d'une longueur donnée ou de longueur maximale, présents, selon une des définitions de ressemblance entre modèle et mots, dans au moins un certain nombre des chaînes. Chacune de ces définitions a donné lieu à un algorithme spécifique. Le principe de base de ces algorithmes est cependant le même et s'appuie sur une formule de récurrence. Celle-ci peut être vue comme une recette de construction des modèles d'une certaine longueur (i.e. d'identification de ses occurrences) à partir de ceux de longueur plus petite, construction qui cesse pour un modèle donné dès que celui-ci ne vérifie plus la condition de présence sur un nombre minimal des chaînes de départ. Malgré cette condition d'arrêt, la nature combinatoire de la construction pose bien sûr un problème important de complexité algorithmique. Nous proposons ici des algorithmes qui sont tous linéaires dans la longueur totale des chaînes, et éventuellement exponentiels uniquement dans le degré de souplesse autorisé dans la ressemblance entre mots et modèles. Ce facteur exponentiel est intrinsèque au problème et représente toujours une situation de pire cas. Observons au passage que, bien que nous n'ayons parlé jusqu'à présent que de l'analyse d'un ensemble de plusieurs objets, il est clair que n'importe lequel de ces algorithmes peut être utilisé afin de rechercher des mots qui apparaissent répétés dans une seule chaîne.

Il nous a semblé important que ce travail présente également une synthèse de toute une longue série de travaux effectués avant nous sur ce même thème, ou sur des thèmes proches. Il y a deux raisons à cela. La première est que nos définitions et algorithmes se sont souvent inspirés de ceux élaborés par d'autres et nous voulions replacer les notres dans leur contexte. La seconde raison est que nous pensons qu'une étude aussi complète que possible de tout ce qui a été fait dans un domaine, que ce soit au niveau des concepts que de celui des algorithmes, représente une base solide et nécessaire avant de partir vers des chemins complètement nouveaux. Notre

contribution à ce domaine a été à la fois :

1. de rendre beaucoup plus souples les définitions de ressemblance entre mots dans les chaînes;

2. de formaliser de manière plus précise toutes ces définitions;

3. enfin, de conjuguer comparaison multiple de chaînes macromoléculaires avec une exploration combinatoire exacte de l'espace de ces chaînes d'une façon qui est relativement efficace en pratique.

Ce travail représente donc à la fois une continuation et une innovation par rapport à ce qui existait déjà.

Le texte est divisé en six chapitres. Le premier offre une introduction générale au problème biologique. Bien que nullement complète, elle essaye de donner un aperçu assez approfondi de tout ce qui peut motiver la recherche que nous réalisons ici. La première partie de cette introduction décrit ainsi les mécanismes biochimiques fondamentaux tandis que la seconde est plus tournée vers les motivations de ce travail. Le chapitre 3 poursuit cette étude du problème biologique puisque nous y réalisons une exploration de la notion de ressemblance, entre monomères d'abord, puis entre chaînes et mots dans ces chaînes. C'est dans les deux dernières sections de ce chapitre que nous introduisons les diverses définitions de ressemblance. Nous y présentons non seulement les notres mais également celles qui ont été proposées avant nous et sur lesquelles nous nous sommes souvent appuyés. À la base de toutes ces définitions se trouve en fait un même concept, qui est celui d'un alignement. Ce dernier peut être considéré comme une mise en correspondance de divers objets en vue de les comparer. Là où nos définitions diffèrent de celles établies antérieurement est que les notres utilisent, comme nous l'avons vu, la notion d'un modèle. Les mots dans les chaînes sont donc comparés de manière indirecte, alors que dans les autres cas, cette comparaison est directe. Les méthodes de comparaison directe d'objets, chaînes ou mots dans les chaînes, sont présentées dans le chapitre 4. C'est là que nous réalisons une synthèse des travaux effectués dans ce domaine avant nous. Puis, dans le chapitre suivant, nous introduisons nos propres méthodes indirectes. Ce chapitre contient ainsi une description détaillée des divers algorithmes que nous avons élaborés, chacun permettant d'identifier les modèles présents au moins un certain nombre de fois dans les chaînes d'un ensemble. Ces deux chapitres sont plus purement informatiques, et ce n'est que dans le cinquième que nous revenons à la biologie puisque ce chapitre est consacré à des illustrations. Nous y présentons divers résultats obtenus avec nos méthodes sur un certain nombre d'exemples. La plupart de ceux-ci ont été choisis dans la littérature et sont considérés comme 'très difficiles'. Notre intention en testant sur eux nos algorithmes était de mesurer leur efficacité, robustesse et intérêt. Ce chapitre comporte ainsi une discussion de ce que ces algorithmes peuvent faire, mais également de ce qu'ils ne sont pas capables pour l'instant de réaliser, ainsi que les éventuelles difficultés liées à leur utilisation. Cette discussion ouvre la voie vers les perspectives offertes par ce travail. Notre sentiment est que celui-ci doit se transformer et aller beaucoup plus loin. En particulier, l'impression, mais elle est très personnelle, est que certaines notions qui sont depuis longtemps acceptées dans le domaine, et que nous-mêmes avons adoptées ici (comparaison multiple, distance d'édition entre chaînes etc) ne sont pas encore totalement mûres ou parfaitement réfléchies. Certaines sont peut être même inadéquates. Le chapitre 7 est consacré alors à une remise en cause de quelques-unes de ces notions. Nous suggérons également quelques voies nouvelles possibles.

Observons pour terminer que ce texte long a été conçu pour des personnes de formations et de goûts divers, et organisé en ce sens dans la mesure du possible. Ainsi, une personne intéressée

uniquement par l'aspect biologique du problème consultera essentiellement le chapitre 2, le chapitre 3, en particulier les sections 1, 2 et 3 contenant les notions générales, ainsi que le chapitre 6. Le lecteur concerné par la partie informatique peut ne lire que les deux dernières sections du chapitre 3 (Définitions de ressemblance), le chapitre 4 et le chapitre 5. Ceux qui veulent avoir un aperçu général de nos algorithmes, mais ne veulent pas entrer dans le détail de leur fonctionnement, peuvent se contenter de lire la première moitié du chapitre 5. Enfin, le chapitre 7 devrait, nous l'espérons, intéresser tout le monde.

Chapitre 2

Le problème biologique

God is indeed a jealous God
He cannot bear to see
That we had rather not with Him
But with each other play

Poème #1719. Emily Dickinson.
The Complete Poems of Emily Dickinson.
Thomas H. Johnson (Ed.). Faber and Faber, 1975.

Le but de ce chapitre est double. Le premier est de présenter un schéma des principes de base de la mécanique de la vie ainsi que ses constituants fondamentaux. L'idée n'est pas de faire de cette première partie un cours de biologie moléculaire. Des ouvrages excellents et bien plus complets peuvent être trouvés qui remplissent cette fonction, certains de nature didactique [Creighton, 1993] [Daune, 1993] [Freifelder, 1990] [Kruh, 1987] [Lewin, 1994] [Li and Graur, 1991] [Russell, 1992] [Stryer, 1981] [Watson *et al.*, 1987] d'autres de réflexion et donc parfois plus polémiques [Danchin, 1990] [Jacob, 1970] [Jacob, 1981] [Jacob, 1987] [Kaufmann, 1993] [Lorenz, 1975] [Monod, 1970] [Nei, 1987] (le choix ici est personnel : ce sont les livres qui ont été lus ou relus durant le développement de ce travail). L'intention de cette première partie est plutôt de fournir les quelques notions qu'il faut avoir du sujet, d'abord pour situer le problème qui va nous concerner, ensuite pour introduire le vocabulaire qui va être utilisé et attirer l'attention sur les points importants, enfin surtout pour commencer à explorer certaines questions intéressantes que ce sujet soulève. Quelques-unes de ces questions sont commentées dès la fin de cette première partie. Si leur présentation paraît parfois confuse, cela est un reflet de leur complexité, qui entraîne souvent une difficulté à évaluer de façon claire leur légitimité et à en dégager les aspects essentiels. Il est néanmoins important d'essayer de les poser.

Ces questions sont reprises dans le début de la deuxième partie dont le but, le second de ce chapitre, est d'essayer de les caractériser de manière plus abstraite, puis de préciser quel problème a été choisi pour être traité dans le reste de ce travail. Les raisons de ce choix sont multiples, souvent d'ordre purement pragmatiques, et un premier commentaire sur ses avantages et limites clôt ce chapitre. Ce commentaire ainsi que les autres questions laissées de côté sont repris et approfondis plus tard, une fois qu'auront été présentées et discutées diverses façons de traiter de manière algorithmique la question qui a été sélectionnée, façons qui ont été élaborées dans le cadre de ce travail ou par d'autres auparavant.

2.1 Les mécanismes de la vie

2.1.1 Principes de base

2.1.1.1 Transmission de l'information

On appelle matériel génétique le support chimique de l'information transmise d'une génération à une autre chez les êtres vivants. Ce matériel, qui flotte librement dans la cellule (c'est le cas des organismes appelées des procaryotes) ou qui se trouve contenu à l'intérieur d'un noyau (c'est le cas des organismes appelés des eucaryotes) a une double nature : une partie représente l'information qui permet de créer les agents exécutants de la machinerie de la vie, une autre concerne les signaux qui permettent de contrôler cette machinerie. Ce contrôle est nécessaire, d'une part pour répondre rapidement aux modifications des conditions de l'environnement dans lequel se déroule la vie de l'organisme, d'autre part pour des raisons de coordination. Si le premier besoin est plus important chez les procaryotes et les eucaryotes unicellulaires car ils sont plus dépendants des variations du milieu extérieur, le second est essentiel à tous les organismes.

À l'exception de certains virus, ce matériel génétique est toujours stocké dans l'ADN et les agents exécutants de la machinerie de la vie sont essentiellement les protéines. Celles-ci sont obtenues de l'ADN de manière indirecte, leur synthèse se faisant en réalité à partir d'un autre agent moléculaire, l'ARN, dont le rôle précis dépend de son type. L'ARN peut aussi exercer une fonction de contrôle de certains processus liés à cette synthèse. Nous reviendrons là-dessus plus loin.

ADN et ARN sont communément appelés les acides nucléiques et représentent avec les protéines les deux principales macromolécules chimiques des organismes vivants, la troisième étant les polysaccharides.

Ces deux familles de macromolécules sont de longs polymères linéaires synthétisés à partir de petites molécules, appelées monomères, qui sont les nucléotides pour les acides nucléiques et les acides aminés pour les protéines.

Les nucléotides sont constitués par l'association d'une base azotée, d'un sucre qui est le ribose pour l'ARN et le désoxyribose pour l'ADN, et d'un groupement phosphate (voir figure 2.1). Chaque nucléotide est rattaché au nucléotide suivant sur la chaîne, appelée chaîne nucléique, par un lien sucre-phosphate (liaison phospho-diester). Les nucléotides sont au nombre de quatre dans l'ADN, distincts entre eux par la structure de leur base azotée. Les noms de ces bases et les symboles qui servent à les dénoter sont indiqués dans le tableau 2.1. Ces bases forment deux classes, celle des purines et celle des pyrimidines. Ce tableau montre aussi les quatres types de nucléotides présents dans l'ARN. Trois comportent les mêmes bases que ceux de l'ADN, un quatrième a pour base l'Uracile en remplacement de la Thymine mais les propriétés chimiques des deux sont très proches.

Les acides aminés quant à eux sont constitués par un groupement amino NH_2, un groupement acide carboxyl COOH, un atome d'hydrogène et une chaîne latérale représentant un radical organique, le tout lié à un atome de carbone appelé carbone α (voir figure 2.2). Le lien qui unit un acide aminé au suivant le long de la chaîne s'appelle liaison peptide et la chaîne elle-même est appelée chaîne polypeptidique. Les acides aminés sont au nombre de vingt. Ils sont caractérisés par leur chaîne latérale. Leur nom, le symbole servant à les dénoter ainsi que leurs principales propriétés physico-chimiques sont donnés dans le tableau 2.2. On emploie aussi le terme de résidu pour désigner un acide aminé dans une chaîne.

À cause des liaisons chimiques diverses qui peuvent s'établir entre les monomères, nucléotides ou acides aminés, d'une macromolécule, celle-ci adopte en outre une structure dans l'espace

Figure 2.1: Un nucléotide.

Nom	Code
Adénosine	A
Thymine	T
Cytosine	C
Guanine	G
Uracile	U

Tableau 2.1: Les nucléotides.

Figure 2.2: Un acide aminé.

Nom	Code à 3 lettres	Code à 1 lettre	Principales propriétés
Alanine	Ala	A	très petit, hydrophobe
Arginine	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	hydrophobe, aromatique
Proline	Pro	P	petit
Sérine	Ser	S	très petit, polaire
Thréonine	Thr	T	petit, hydrophobe, polaire
Tryptophane	Trp	W	hydrophobe, aromatique, polaire
Tyrosine	Tyr	Y	hydrophobe, aromatique, polaire
Valine	Val	V	petit, aliphatique, hydrophobe

Tableau 2.2: Les acides aminés.

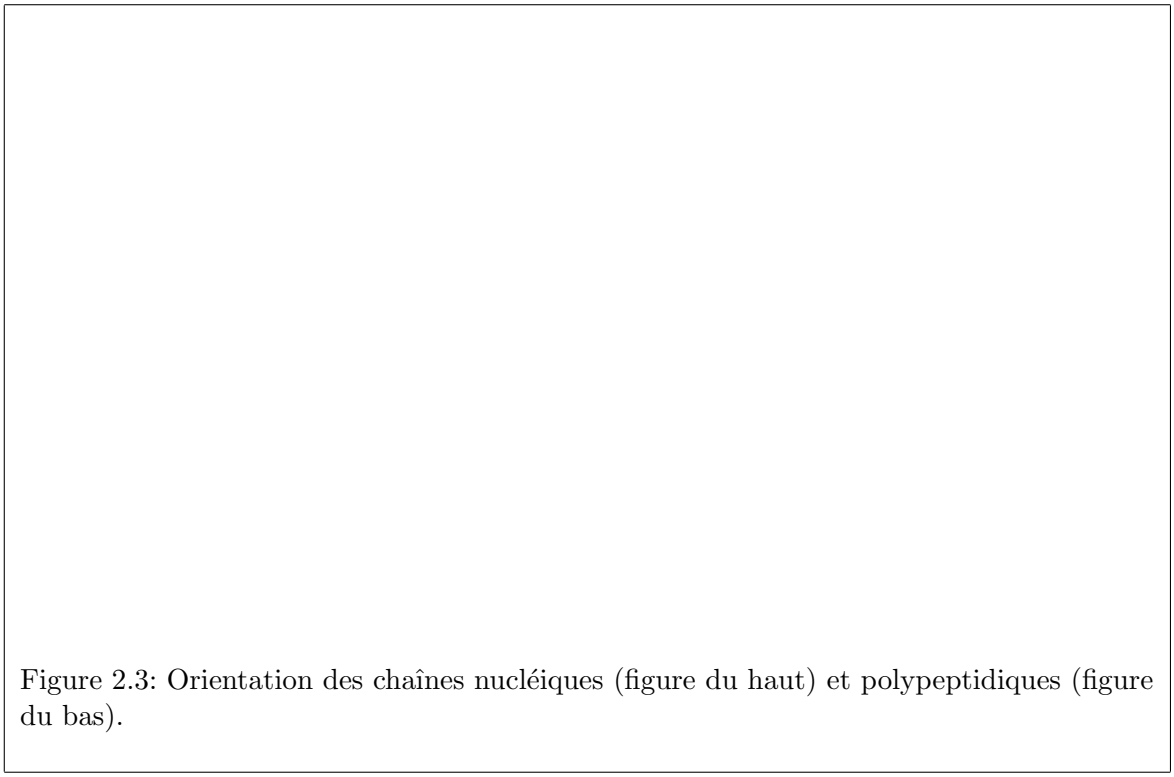


Figure 2.3: Orientation des chaînes nucléiques (figure du haut) et polypeptidiques (figure du bas).

qui lui est très spécifique. L'asymétrie des liaisons phospho-diester et peptidique implique par ailleurs que les chaînes nucléiques aussi bien que polypeptidiques possèdent une orientation. Celle-ci va de l'extrémité 5' (phosphate) à l'extrémité 3' (sucre) pour les acides nucléiques et du groupement amino dont l'amine N est libre au groupement carboxyl dont le carbone est libre pour les protéines (voir la figure 2.3). Pour l'ADN, la conformation spatiale n'est pas aussi complexe que pour les protéines et, dans une moindre mesure, pour certains ARN mais le rôle qu'elle joue est néanmoins important.

L'ADN est ainsi formé de deux chaînes qui s'enroulent l'une autour de l'autre pour former la structure bien connue en double hélice [Watson and Crick, 1953]. Les deux chaînes sont maintenues solidaires par des liaisons hydrogène entre les bases des deux brins où, pour des raisons stéréochimiques, l'Adénine (A) s'associe toujours avec la Thymine (T) et la Cytosine (C) avec la Guanine (G). Adénine et Thymine d'un côté, Cytosine et Guanine d'un autre sont appelées des bases complémentaires. La synthèse de nouvelles molécules d'ADN procède par la séparation de ces deux brins, suivie de la reconstitution nucléotide par nucléotide de deux autres brins complémentaires. Cette reconstitution donne lieu à la création de deux nouvelles molécules d'ADN, chacune possédant un brin de la molécule initiale (voir figure 2.4). Ce processus semi-conservatif est appelé la réplication.

La synthèse des molécules d'ARN est obtenue de façon similaire à partir d'une molécule d'ADN, sauf que cette fois une seule des chaînes de l'ADN sert de matrice pour la formation d'un unique brin d'ARN et que la base complémentaire de l'Adénine est l'Uracile. Cette synthèse constitue le processus de la transcription.

Trois types d'ARN sont ainsi produits, distincts par leurs structures et par le rôle qu'ils peuvent jouer dans la formation de nouvelles protéines. L'ARN messager sert de matrice dans la synthèse de ces dernières, les ARN de transfert se combinent aux acides aminés qu'ils amènent

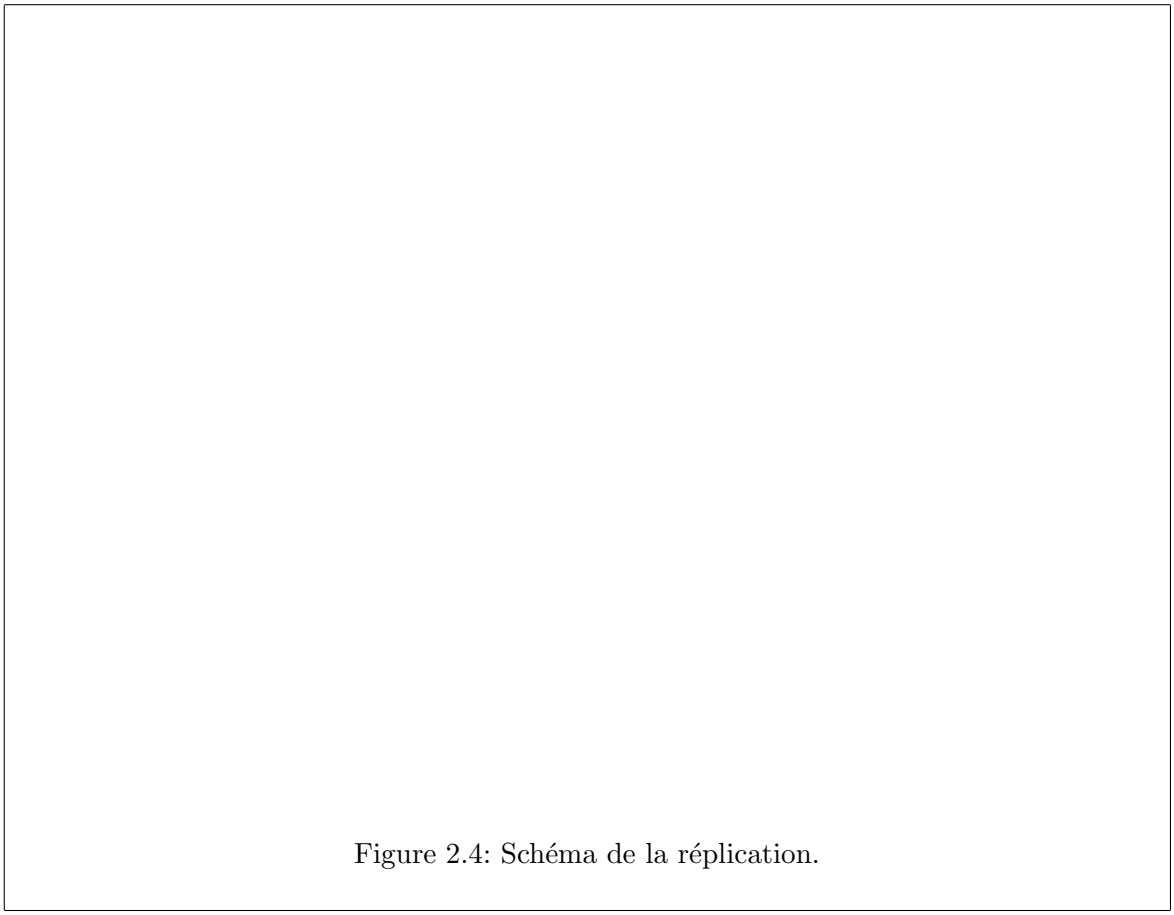


Figure 2.4: Schéma de la réplication.

1 — 2	U	C	A	G	3
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	Non sens	Non sens	A
	Leu	Ser	Non sens	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Tableau 2.3: Le code génétique.

sur la matrice pour être polymérisés en protéines et finalement les ARN ribosomaux assistent l'ARN messager dans ce processus de synthèse qui est appelé la traduction.

La séquence d'ARN messager est en relation 3:1 avec la séquence d'acides aminés qui est ainsi obtenue : une suite de trois nucléotides code pour un acide aminé. On appelle un tel triplet de nucléotides un codon. Des 64 triplets possibles, 3 représentent des signaux de terminaison de la traduction et 61 codent pour 20 acides aminés. Ce code, donné dans le tableau 2.3 est donc dégénéré mais non ambigu dans le sens ARN→protéines. Il est aussi le même dans pratiquement tous les organismes (des exceptions ont néanmoins été relevées chez les mitochondries, les protozoaires et les mycoplasmes).

Ce qu'il faut observer ici c'est que tout l'ADN n'est pas transcrit en ARN, et que tout l'ARN messager n'est pas traduit en protéines. Les parties de l'ADN qui sont transcrites et qui correspondent à ce qui est appelé les gènes peuvent représenter un pourcentage plus ou moins grand suivant les organismes (il est en général élevé chez les procaryotes et petit chez les eucaryotes). Certaines parties n'ont aucune fonction apparente mais d'autres régions qui ne sont pas transcrites jouent malgré tout un rôle très important dans le contrôle à l'échelle moléculaire. Ce point sera vu plus loin. Il en est de même pour certaines régions de l'ARN messager qui ne sont pas traduites en protéines. Ces régions peuvent se situer avant ou après la partie qui sera transcrite ou traduite, elles peuvent également se trouver dispersées au milieu. Cela est le cas pour les ARN messagers des eucaryotes qui présentent en alternance des zones, appelées exons, codant pour des protéines et des zones, appelées introns, dont l'origine et la fonction ne sont pas bien connues et qui sont excisées de l'ARN avant que celui-ci soit traduit (ce phénomène est appelé l'épissage, voir figure 2.5). Par contre, il peut y avoir chevauchement

Figure 2.5: Schéma de la transcription et épissage.

de l'information chez les ARN messagers de certains procaryotes où un gène, ou une partie d'un gène, peut servir à la synthèse de plus d'une protéine. Chez ces organismes, la superposition de l'information peut aussi affecter les régions non transcrites ou non traduites qui participent néanmoins au contrôle de ces processus.

2.1.1.2 Le fonctionnement de la 'machine chimique'

C'est par le terme de 'machine chimique' que Jacques Monod [Monod, 1970] qualifie les organismes vivants. Il s'agit de machines capables de se constituer de manière autonome grâce à des "*interactions constructives internes*" dont les agents essentiels sont les protéines.

Les activités que ces dernières exercent au sein d'un organisme sont principalement de deux types, l'une est métabolique et est assurée par une classe de protéines appelées enzymes, l'autre est cybernétique et est réalisée principalement par les protéines dites régulatrices qui jouent le rôle de détecteurs de signaux chimiques et permettent à l'organisme de s'auto-contrôler. Les protéines possèdent en outre un rôle important de transport d'autres molécules, d'électrons ou de protons à l'intérieur ou vers l'extérieur des cellules. Enfin, elles peuvent avoir un rôle plus purement mécanique de maintien de la structure macroscopique. Cette dernière fonction est assurée par les protéines dites de structure qui sont des molécules très allongées. Les autres fonctions de catalyse et de régulation sont quant à elles réalisées essentiellement par les protéines dites de fonction qui sont de loin les plus nombreuses. Ce sont des macromolécules qui se replient sur elles-mêmes de façon extrêmement complexe donnant lieu à une structure généralement très compacte.

Ce qu'il est important d'observer pour l'instant est que toutes les activités chimiques de ces protéines reposent en fait sur leurs propriétés stéréochimiques, c'est-à-dire sur leurs capacités à reconnaître d'autres molécules et que cette capacité est intimement liée à la configuration que ces protéines adoptent dans l'espace.

C'est ainsi à travers une reconnaissance d'"*aires complémentaires*" [Monod, 1970] qu'une protéine se lie à une autre molécule pour former un complexe chimique. La nature de cette

complémentarité n'a pas encore été déterminée. Parmi les images qui ont été utilisées, on trouve celle du modèle clé-serrure ou modèle à deux états [Monod *et al.*, 1963] [Monod *et al.*, 1965] ou celle d'un ajustement induit ou modèle séquentiel [Koshland *et al.*, 1966] (voir les figures 2.10 et 2.11). D'autres conceptions plus 'dynamiques' ont été également proposées et font intervenir soit des changements d'états de la structure de la protéine, changements considérés comme relativement rigides car limités par des contraintes stéréochimiques, soit des mouvements internes de la macromolécule pouvant être plus fluides et amples [Creighton, 1993] [Daune, 1993] [Karplus and McCammon, 1983] [Karplus *et al.*, 1987] [Gribskov, 1992].

Chez la plupart des protéines, ces zones de reconnaissance ne concernent qu'une partie relativement très petite de la structure. La localisation de ces zones dépent quant à elle du type d'interaction observé. Celle-ci peut impliquer soit deux protéines non-enzymatiques, soit une enzyme et la molécule sur laquelle elle agit appelée substrat, soit encore une protéine et une molécule d'ADN. Dans les trois cas, les aires en contact lors de la formation du complexe peuvent se situer à la surface des macromolécules. Lorsqu'il s'agit par contre d'une enzyme et de son substrat, ou d'une protéine régulatrice venant se fixer sur une molécule d'ADN, l'interaction peut également concerner des régions centrales de la protéine. Le substrat catalysé par les cytochromes P450 par exemple pénètre entièrement à l'intérieur de la protéine à travers une sorte de canal. La reconnaissance se fait alors sans doute dès l'entrée de ce canal, mais la spécificité de l'interaction est probablement réalisée par la zone de contact interne. Les contacts enzyme-substrat ne sont toutefois pas toujours de ce type, et peuvent aussi avoir lieu à la surface de l'enzyme. Cela est le cas en particulier des immunoglobulines et des récepteurs hormonaux. Un contact en surface est celui le plus souvent observé également dans le cas d'une interaction protéine-ADN. Cependant, une partie de la protéine peut parfois venir 'embrasser' la molécule d'ADN (protéine LeuZipper) ou totalement l'enserrer (protéine Topoisomérase). Ces divers types d'interactions sont illustrées dans les figures 2.6, 2.7 et 2.8.

Ces observations sont importantes et nous y reviendrons ultérieurement, ainsi que sur les structures et, surtout, sur la relation chaîne d'acides aminés, structure et fonction. Mais avant cela, nous allons parler un peu des mécanismes de contrôle d'un organisme.

2.1.1.3 Les mécanismes de contrôle

Les mécanismes de contrôle dont dispose un organisme au niveau moléculaire affectent les trois étapes décrites précédemment : réplication, transcription et traduction. Nous ne parlerons ici que de la régulation de ces deux dernières étapes.

Ces mécanismes sont extrêmement complexes et seuls sont présentés les points qui sont indispensables pour comprendre ce que nous sommes amenés à chercher lorsque nous essayons de les analyser au moyen d'algorithmes.

Comme nous l'avons déjà mentionné, le contrôle à l'intérieur d'un organisme ou d'une cellule est essentiel principalement pour deux raisons : la première est qu'il faut qu'il puisse y avoir une réaction rapide à des modifications survenues dans l'environnement externe ou interne, la seconde est que l'organisme doit pouvoir coordonner de façon efficace tous ses moyens et activités. La plupart des contrôles moléculaires font donc référence à deux types d'information : l'une concerne les signaux régulateurs et fait partie du matériel génétique, l'autre provient presque toujours du milieu extérieur et se réfère en général à la présence ou à l'absence d'une certaine substance chimique dans ce milieu [Lewin, 1994]. Cette dernière est une information qui n'est pas directement stockée dans l'ADN au contraire de l'information génétique, mais elle y est cependant préservée d'une façon indirecte puisque les sites sur l'ADN qui reconnaissent la présence de cette information externe sont conservés.

Figure 2.6: Localisation des zones de contact : pénétration du substrat (représenté en noir) à l'intérieur de la protéine (cytochrome $P450_{Cam}$).

Figure 2.7: Localisation des zones de contact : la protéine LeuZipper (les deux hélices en forme de rubans) 'embrasse' la molécule d'ADN.

Figure 2.8: Localisation des zones de contact : la protéine Topoisomérase cristallisée sans son ADN mais dont la forme laisse supposer que la molécule d'ADN vient s'enserrer à l'intérieur de la cavité montrée dans la partie supérieure de la figure (la dimension de cette cavité (28 Angström) correspond au diamètre de la double hélice d'ADN).

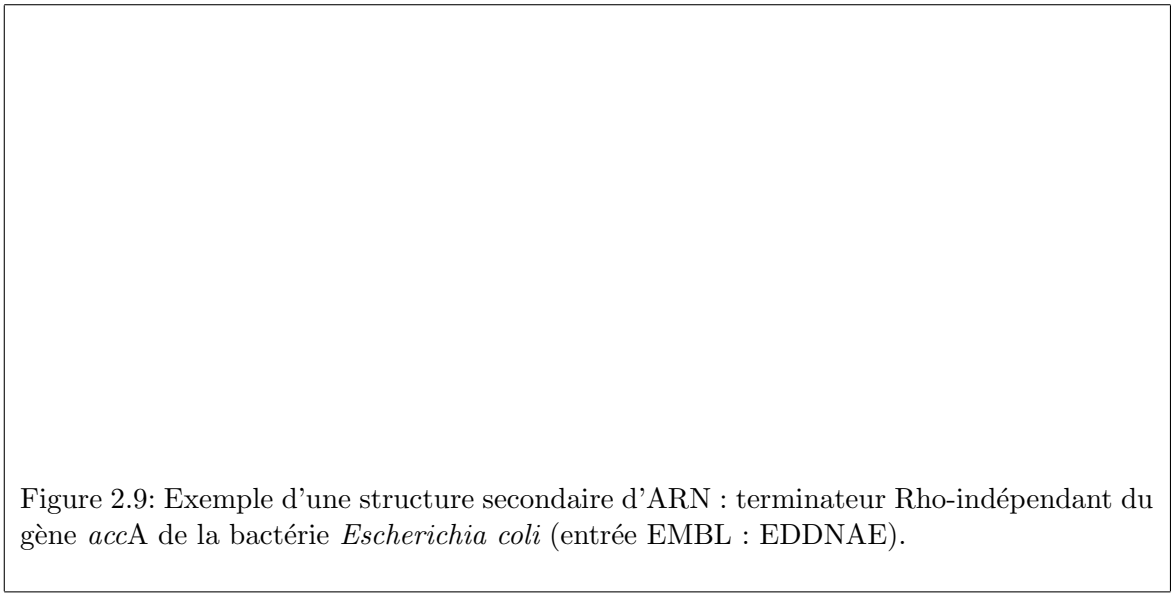


Figure 2.9: Exemple d'une structure secondaire d'ARN : terminateur Rho-indépendant du gène *acc*A de la bactérie *Escherichia coli* (entrée EMBL : EDDNAE).

L'activité de contrôle est réalisée à travers la reconnaissance par une molécule régulatrice d'un signal présent soit sur l'ADN, soit sur l'ARN messager, suivie de la formation (ou de l'empêchement de formation) d'un complexe chimique entre la molécule et la chaîne nucléique. Lorsque la molécule régulatrice est une protéine ou un ARN, la région qui reconnaît, aussi bien que celle reconnue, sont appelées sites.

La nature du site sur l'ADN ou l'ARN représentant un signal régulateur n'est pas toujours connue de manière exacte. Elle peut aussi être très complexe. Ce signal peut ainsi concerner une suite plus ou moins longue de positions contiguës sur ces macromolécules, ou une série de positions discontinues, ou encore une série de suites de plus petites tailles, le signal de chaque sous-suite pouvant moduler l'effet des signaux des suites situées en amont ou en aval sur la chaîne. Il est probable qu'il existe des corrélations à courte ou à longue distance entre les positions. Par ailleurs, d'éventuelles relations de symétrie peuvent impliquer l'existence de propriétés structurales. Ainsi, la reconnaissance du site de terminaison de la transcription de l'ARN messager dans certains organismes se fait par la propre molécule en train d'être transcrite qui adopte une structure spéciale en forme d'épingle à cheveux à l'endroit où la transcription doit prendre fin. Très brièvement, une telle structure peut être décrite comme étant formée d'une tige composée de deux segments complémentaires inversés (palindrome biologique) qui se trouvent appariés, d'une boucle de nucléotides non appariés et éventuellement d'une excroissance située quelque part le long de la tige et constituée de nucléotides également non appariés (voir figure 2.9). D'autres modifications de la conformation d'un ARN messager peuvent faire partie d'un système de contrôle de la fonction exercée par celui-ci. Ces modifications surviennent soit parce qu'une partie de la molécule a été excisée ailleurs, soit parce que l'interaction d'une molécule régulatrice (protéine par exemple) avec un site de l'ARN affecte la structure de ce dernier à un autre site.

La reconnaissance d'un signal de la part des molécules régulatrices, en particulier des protéines, est quant à elle extrèmement liée comme nous l'avons vu à leur structure tridimensionnelle. Le contrôle que les protéines exercent peut impliquer plus d'un site de cette molécule simultanément. Cela est le cas pour la régulation de début de transcription par exemple. Celle-ci peut être négative (la présence de la protéine sur la chaîne nucléique empêche le processus

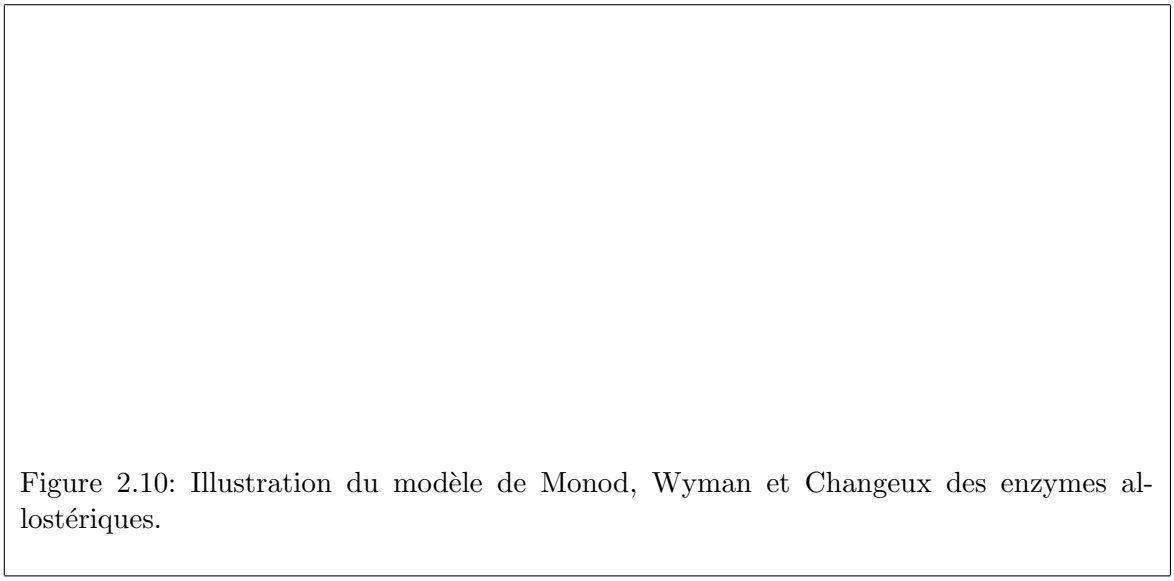


Figure 2.10: Illustration du modèle de Monod, Wyman et Changeux des enzymes allostériques.

de se déclencher) ou positive (sa présence est nécessaire pour que le processus puisse débuter). Il est important de noter ici que, dans les deux cas, l'activité de la protéine régulatrice peut être elle-même modulée par la formation ou non d'un autre complexe avec une molécule plus petite. Cela est le cas des protéines enzymatiques allostériques dont un modèle déjà mentionné a été proposé par Monod [Monod *et al.*, 1963] [Monod *et al.*, 1965] (voir la figure 2.10) et un autre l'a été par Koshland (voir la figure 2.11).

Les enzymes allostériques possèdent ainsi deux types de sites : le site actif qui se combine au substrat et qui catalyse la réaction enzymatique et les sites régulateurs qui peuvent se combiner à un effecteur, soit activateur, soit inhibiteur. Les deux types de sites sont séparés sur la molécule. Dans certaines enzymes ils appartiennent même à des chaînes polypeptidiques différentes. Il existe alors deux états de l'enzyme, un état catalytique et un état inhibé, qui se trouvent en équilibre. Dans l'état catalytique, la forme de l'enzyme lui permet de se combiner à l'activateur et au substrat (effet coopératif). Dans la forme inhibée, seul l'inhibiteur peut se combiner et l'enzyme est inactive. Le passage d'une forme à l'autre est appelée transition allostérique.

Au moins autant que pour certains ARN, la structure d'une protéine est donc essentielle pour comprendre la fonction que celle-ci exerce. La section suivante est consacrée à une rapide description de ces structures puis nous parlerons des facteurs de variabilité à l'échelle moléculaire.

2.1.2 Structures

2.1.2.1 Protéines

2.1.2.1.1 Description de la structure

Plusieurs forces de nature chimique font qu'une protéine nouvellement synthétisée se replie sur elle-même pour adopter une certaine configuration dans l'espace. La structure qui est ainsi créée, et qui est souvent très compacte, est spécifique à la protéine en question. Cela signifie que si une chaîne d'acides aminés identique était synthétisée une seconde fois, elle adopterait la même

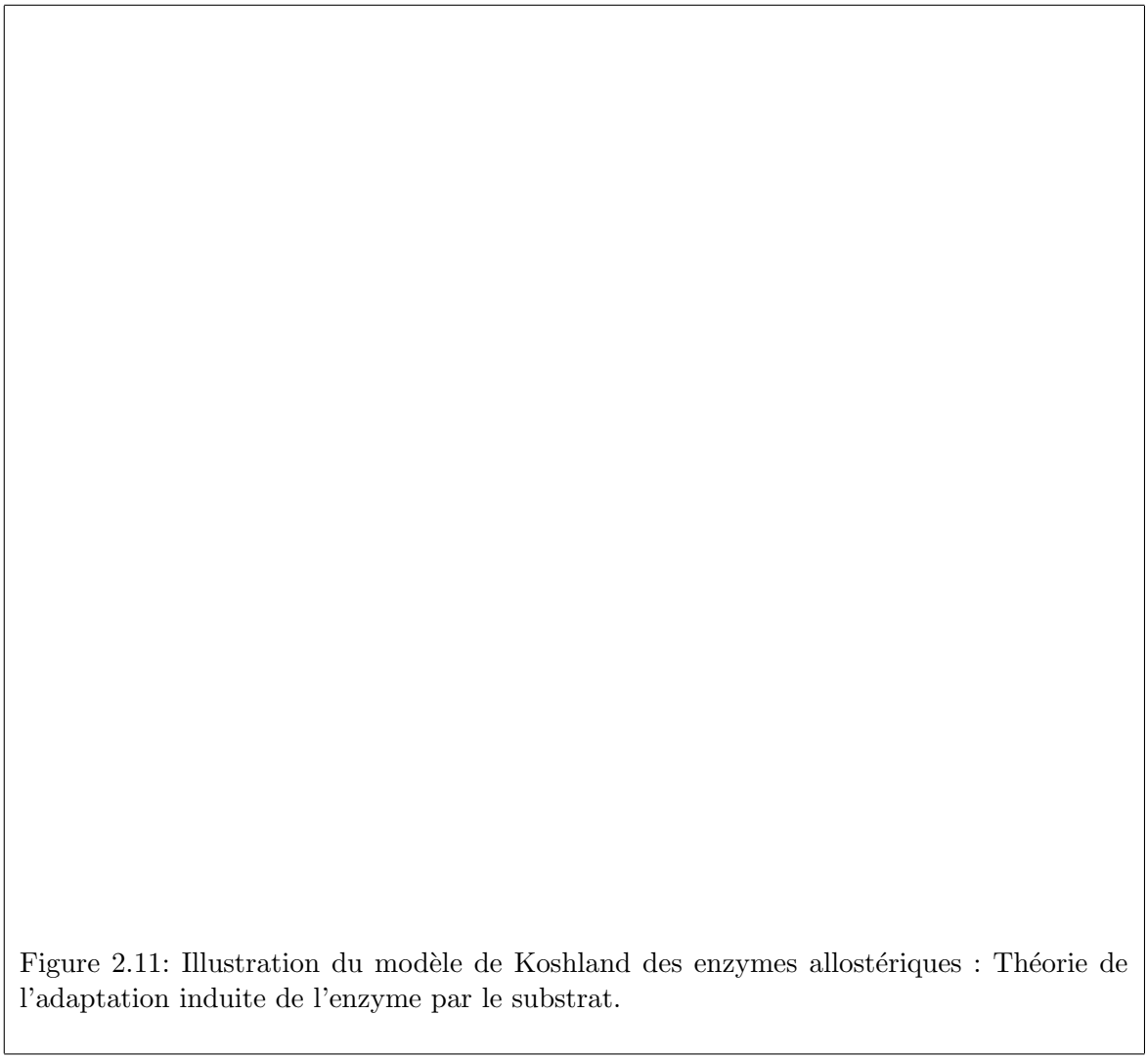


Figure 2.11: Illustration du modèle de Koshland des enzymes allostériques : Théorie de l'adaptation induite de l'enzyme par le substrat.

conformation, ou éventuellement une conformation comportant un très petit nombre d'états distincts, peu distants les uns des autres [Monod, 1970]. Sander [Kabsch and Sander, 1983] et Argos [Argos, 1987] ont montré en effet qu'un certain nombre de pentapeptides (suite de cinq acides aminés d'une chaîne) ne se replient pas toujours exactement de la même façon. La conclusion qu'Argos tire cependant de cette observation est simplement que l'environnement des pentapeptides dans l'ensemble de la protéine joue un rôle essentiel dans leur conformation finale. À ces exceptions près, la relation entre chaîne polypeptidique et structure peut ainsi être considérée comme univoque, mais nous verrons plus loin qu'elle n'est pas bi-univoque. Les lois régissant un tel repliement sont quant à elles mal connues. Il est probable que le repliement commence dès le début de la synthèse de la molécule, avant donc que celle-ci ne soit complètement formée. Une hypothèse qui a également été émise, et qui est de plus en plus acceptée, est que ce phénomène ne se réaliserait pas toujours de manière spontanée mais serait guidé par d'autres protéines appelées des chaperones. Le rôle de ces dernières ne consisterait pas à catalyser le repliement des protéines, mais plutôt à faciliter l'établissement des liens conduisant à une conformation 'correcte' des macromolécules (en général correspondant à celle d'énergie minimale) ainsi qu'à stabiliser et protéger les étapes intermédiaires du processus de repliement [Clarke, 1986] [Gething and Sambrook, 1992]. Elles réaliseraient éventuellement aussi un travail d"édition' dans la mesure où elles seraient capables de reconnaître et de dégrader les protéines mal formées [Creighton, 1993].

La structure d'une protéine peut être décrite à plusieurs niveaux de complexité spatiale.

Le plus simple se réfère à la suite des acides aminés composant la chaîne. Ce niveau est appelé la structure primaire d'une protéine.

Une protéine peut posséder par ailleurs une certaine régularité spatiale locale et ces conformations, impliquant une seule région ou un nombre réduit de régions, forment la structure secondaire d'une protéine. Les principales structures locales possibles sont les hélices α, les feuillets β et les coudes (voir figure 2.12 et 2.13).

La structure dite tertiaire concerne la conformation totale d'une protéine, résultante de toutes les interactions entre acides aminés de la chaîne, que ceux-ci soient par ailleurs impliqués ou non dans une structure secondaire (voir figure 2.14).

Un niveau intermédiaire entre la structure secondaire et la tertiaire, appelé structure supersecondaire, est parfois employé pour décrire certains assemblages de diverses structures secondaires.

Enfin, certaines protéines sont formées de l'aggrégation de chaînes polypeptidiques qui peuvent être identiques ou distinctes. On les appelle des protéines multimériques, et la structure quaternaire fait référence à l'arrangement spatial de cette aggrégation. Chaque chaîne forme une sous-unité de la structure globale, et obéit à des lois de repliement qui peuvent être indépendantes de celles gouvernant le repliement global de la protéine multimérique.

La composition en plusieurs chaînes n'est pas la seule division qu'il est possible de rencontrer dans une protéine. Même celles constituées d'une seule chaîne et qui possèdent plus de quelques centaines d'acides aminés paraissent être composées en fait de deux, ou plus de deux unités de structure appelées des domaines [Creighton, 1993]. Les domaines d'une protéine peuvent interagir entre eux mais apparemment le font de manière moins étendue que les unités de structure situées à l'intérieur d'un même domaine. L'origine de ces domaines est quant à elle mal déterminée.

2.1.2.1.2 Importance de la structure

Plusieurs observations peuvent être faites à ce point.

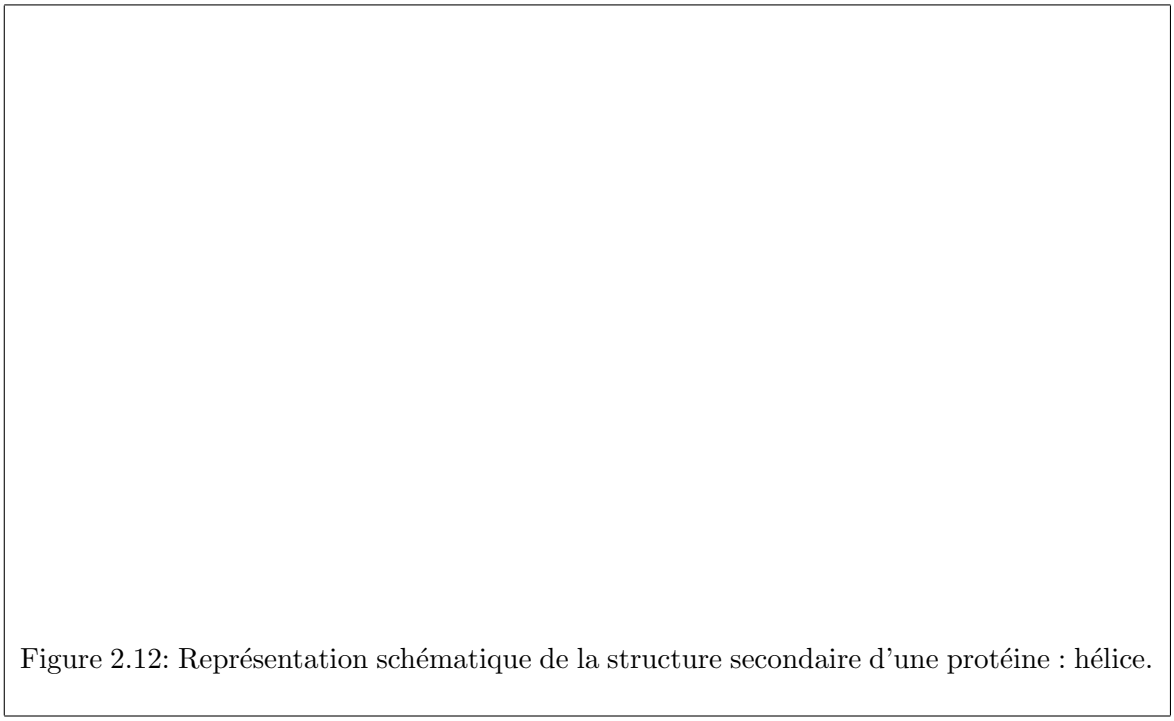


Figure 2.12: Représentation schématique de la structure secondaire d'une protéine : hélice.

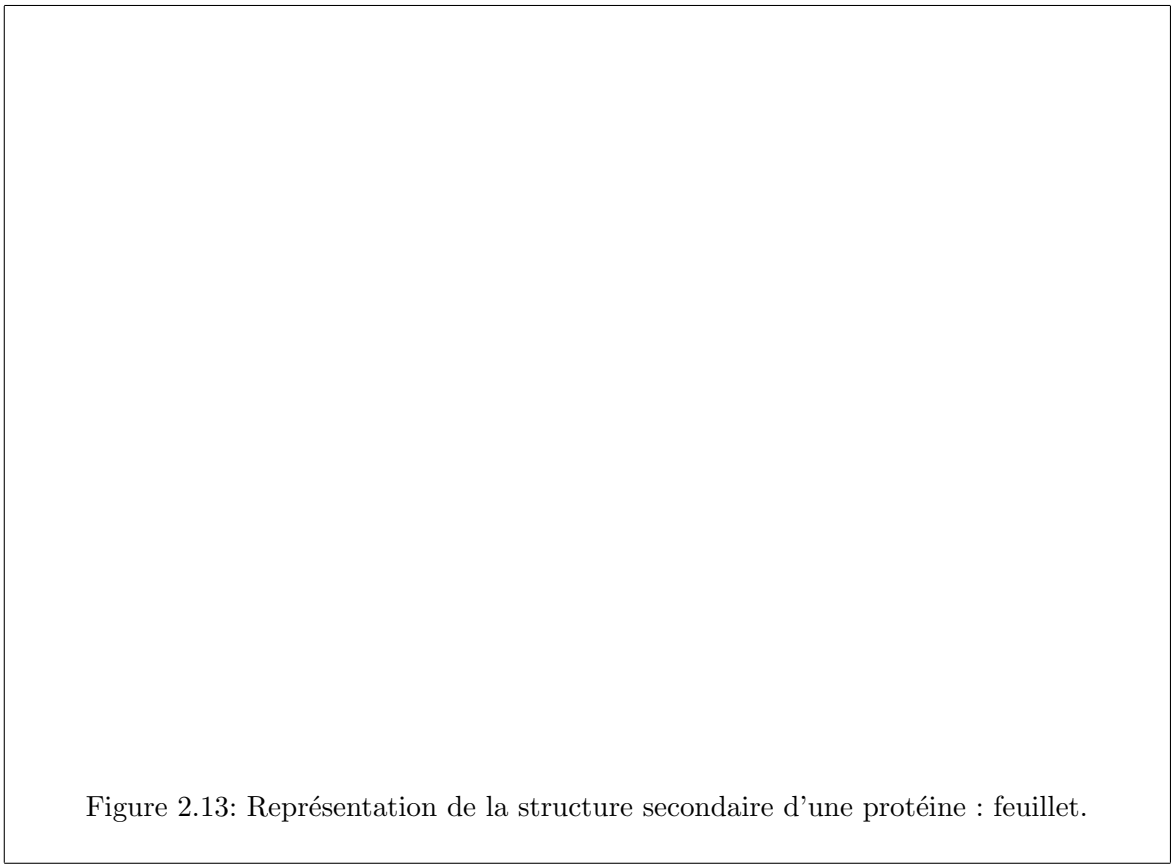


Figure 2.13: Représentation de la structure secondaire d'une protéine : feuillet.

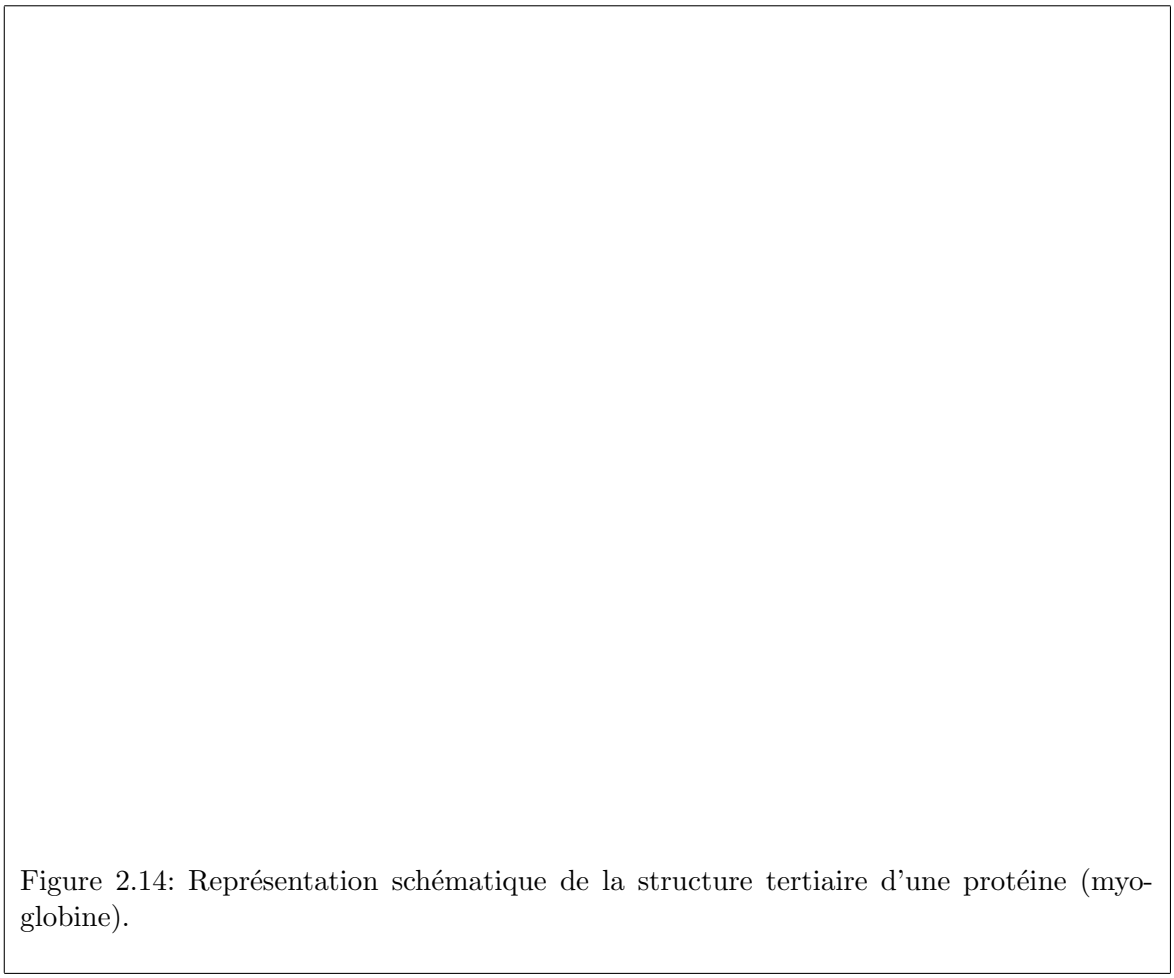


Figure 2.14: Représentation schématique de la structure tertiaire d'une protéine (myoglobine).

La première mentionnée auparavant est que la relation séquence (structure primaire) et structure (conformation spatiale globale) d'une protéine est univoque mais qu'elle n'est pas bi-univoque : deux séquences différentes peuvent ainsi adopter une même structure. Ce fait peut signifier au moins trois choses : soit tous les acides aminés d'une chaîne polypeptidique ne sont pas déterminants pour la structure que celle-ci adopte, ou bien ce qui compte n'est pas l'identité des acides aminés mais tout ou partie de leurs propriétés physico-chimiques, ou encore ce ne sont pas des acides aminés isolés mais des rapports entre acides aminés qui gouvernent la formation de la structure. Bien sûr, il peut aussi y avoir un mélange de ces trois facteurs ou l'existence d'autres encore.

La seconde observation concerne le site de la molécule qui va interagir avec une autre pour réaliser une opération catalytique ou régulatrice (dans le cas des protéines, ce site est appelé site actif). Nous avons vu que ce site n'implique le plus souvent que quelques éléments de la chaîne et que, surtout, il s'agit d'un site structural. C'est la forme qui est la plus importante. Bien sûr cette forme dépend de la composition mais nous venons de voir que la relation séquence-structure n'est pas bi-univoque. Le fait que ce site représente un segment souvent petit de la protéine, qui en outre doit posséder une structure très précise, peut impliquer néanmoins que la nature des acides aminés à ces positions ne puisse varier autant qu'à d'autres endroits sans que la structure locale change et que la fonction soit perdue. Par ailleurs, ce site peut, dans certains cas, concerner des positions qui sont proches dans l'espace mais très éloignées sur la chaîne polypeptidique. Un exemple est celui de la chymotrypsine où seuls trois acides aminés distants l'un de l'autre dans la structure primaire se rencontrent dans la structure tertiaire pour former le site de la protéine. Il faut noter que, dans ce cas, la substitution de l'un quelconque de ces trois acides aminés abolit l'activité catalytique de la molécule. Par contre, dans les protéines qui s'attachent à une chaîne nucléique, le site actif concerne des régions souvent plus étendues. Enfin, chez certaines protéines, en particulier les protéines de structure, c'est toute la structure qui est impliquée dans la fonctionnalité de la molécule.

Une dernière observation que l'on peut faire est, comme nous l'avons vu, que les sites actifs se situent tantôt à la surface de la structure protéique [Creighton, 1993], souvent dans les régions peu structurées appelées boucles [Rooman *et al.*, 1992], tantôt à l'intérieur des protéines. Or c'est dans ce noyau de la structure globale et dans les structures secondaires que la régularité des liaisons ou l'entassement des atomes laissent supposer que les exigences chimiques et stériques relatives aux acides aminés présents à ces positions sont les plus strictes. Ce sont aussi ces régions qui sont sans doute les plus déterminantes de la forme spatiale finale que va adopter la macromolécule.

Il est donc important de noter, avant de clore cette section, que les régions qui sont impliquées dans la fonction d'une protéine, et celles qui sont impliquées dans la formation de la structure, ne sont souvent pas les mêmes. Cela se trouve confirmé par le fait que l'on ait pu observer deux protéines ayant des structures différentes mais une même fonction [Bajaj and Blundell, 1984], ainsi que deux protéines ayant un même repliement global mais des fonctions différentes [Chothia, 1984] [Finkelstein *et al.*, 1993], où par un 'repliement global' on entend les mêmes structures secondaires placées dans le même ordre les unes par rapport aux autres ainsi qu'une topologie générale semblable mais pas nécessairement exactement la même structure [Chothia, 1992].

2.1.2.2 Acides nucléiques

Nous n'allons pas entrer dans le détail des structures possibles de l'ARN, ou de celles de l'ADN dont la double hélice peut également se replier dans l'espace.

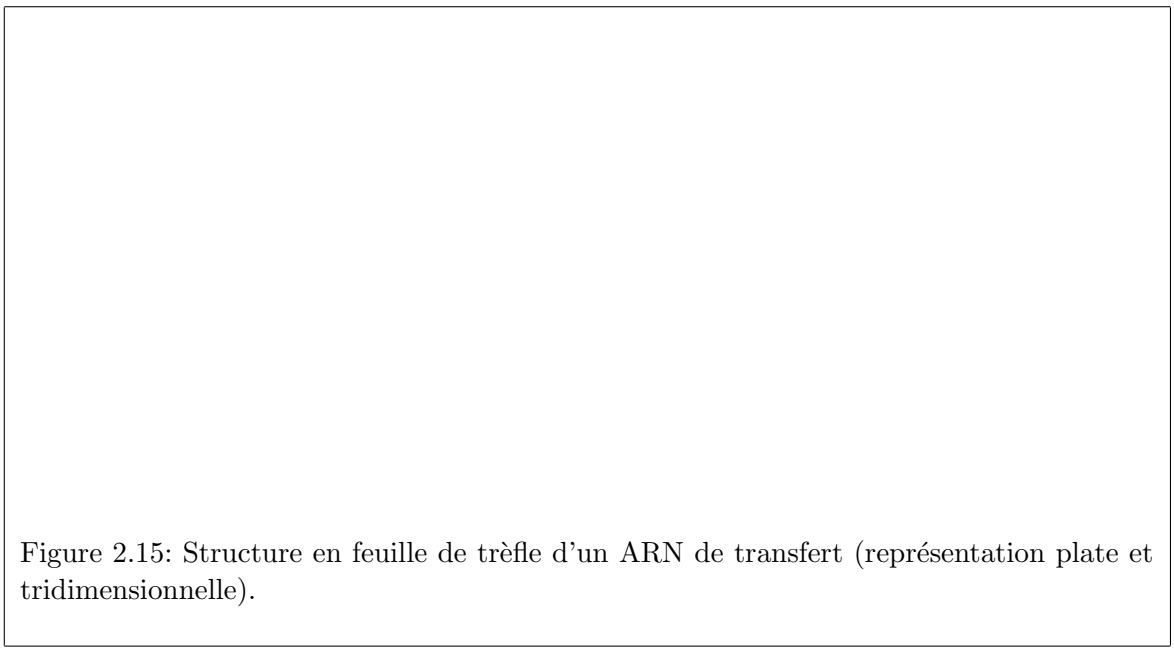


Figure 2.15: Structure en feuille de trèfle d'un ARN de transfert (représentation plate et tridimensionnelle).

Observons toutefois qu'en ce qui concerne l'ARN, les structures secondaires impliquant des nucléotides proches sur la chaîne ribonucléique sont considérées en général comme plus simples que les structures secondaires adoptées par les protéines, dans la mesure où les règles régissant leur formation sont plus faciles à déterminer. C'est en effet principalement par une combinatoire de palindromes biologiques que se créent les diverses structures qui ont pu être observées : en épingle à cheveux comme dans le cas du site de terminaison des ARN messager, en feuille de trèfles (ARN de transfert — voir la figure 2.15) etc.

Ce sont des relations plus complexes entre palindromes qui donnent alors le plus souvent naissance à des formes spatiales tridimensionnelles dans l'ARN. Un exemple est celui des pseudo-nœuds impliquant deux palindromes croisés. Ces formes sont quant à elles au moins aussi compliquées que celles observées dans la structure tertiaire des protéines.

Notons finalement qu'ici, comme dans le cas des protéines, la relation séquence nucléique et structure est univoque mais non bi-univoque. En particulier, il est facile de voir que le simple échange des positions de deux bases appariées sur une tige est déjà une source de variation qui peut ne pas affecter la structure (elle l'affectera si, pour des raisons chimiques, le changement de paire de bases complémentaires rend la structure moins stable que d'autres possibles dans la même région). En règle générale, il n'est donc pas plus aisé que pour les protéines de déterminer la structure secondaire ou tertiaire d'un ARN (ou d'un ADN) à partir de sa séquence.

2.1.3 Facteurs de variabilité

2.1.3.1 Dynamique et évolution : deux temps différents

Dans tout ce qui a été présenté jusqu'à maintenant, le facteur temps n'est jamais intervenu.

Il y a en fait deux temps en biologie, l'un qui est observable de manière directe, et l'autre qui ne l'est que de façon indirecte.

Le premier est relatif à tout changement pouvant intervenir dans une macromolécule, durant le temps de son existence. En général, les changements observables dans une même

molécule sont des changements de nature spatiale. Un exemple classique est celui des protéines allostériques qui, comme nous l'avons vu, peuvent pendant le déroulement de leur activité subir une déformation plus ou moins grande de leur structure. Une molécule est ainsi considérée comme un objet dont les mouvements peuvent jouer un rôle non négligeable dans la fonction qu'elle exerce [Daune, 1993].

Le temps qui va nous intéresser est un temps qui ne concerne plus une seule entité mais un ensemble d'entités. Il ne mesure pas ce qui se passe au niveau d'un objet, ou comment cet objet bouge pendant une certaine période d'observation (qui ne peut être plus longue que le temps de vie de l'objet observé, ou que celui de l'observateur), mais, en quelque sorte, l'état relatif d'avancement des objets les uns par rapports aux autres à un moment donné où le terme avancement ne fait référence à aucune valeur qui pourrait être positive (ou négative) et où les différences portent non plus sur la forme des objets comme dans le cas précédent, mais sur leur composition chimique.

Dans le premier cas, l'objet étudié est unique et le temps représente un intervalle. Dans le second cas, le temps est figé et l'intervalle porte sur les objets. Si le passage du temps est observable malgré tout c'est que l'on imagine — ou bien alors c'est ce que l'on veut justement montrer — que ces objets ont une même origine, un ancêtre commun, dont ils sont tous issus et dont ils se sont tous écartés pas toujours de la même façon. Si l'on dit que l'observation de ce temps est indirecte, c'est que l'on ne 'voit' pas en fait les étapes intermédiaires de son action. Seuls sont connus les effets présents, ceux qui sont observables 'maintenant', tous les autres doivent en être déduits. En outre, ce qui rend la chose plus compliquée est que l'action du temps ne peut pas être mesurée par rapport à un même point référentiel, l'ancêtre commun n'étant pas connu.

Les objets en question sont bien sûr toujours les macromolécules, et ces liens, s'ils existent, sont de nature génétique. Les macromolécules subissent en effet des variations au cours de leur vie et de la vie des organismes au sein desquelles elles vivent et se transmettent, et les effets de la variation sont différents entre deux macromolécules de même nature et de même origine. Le temps qui va nous concerner ici est le temps de l'évolution dont l'échelle est bien plus longue que celle du temps de vie d'une molécule. Ce qui va nous intéresser c'est de découvrir parmi un groupe de macromolécules ce qui a changé et comment cela a changé au cours de ce temps. Les raisons de cet intérêt sont revues plus loin, mais avant nous allons examiner quels sont les facteurs qui peuvent introduire une variation au niveau moléculaire.

2.1.3.2 Facteurs de variabilité

La principale source de variation moléculaire sont les mutations. Une mutation est tout simplement un changement survenu à un instant donné au niveau de l'ADN des cellules germinales de l'organisme (et qui est donc transmissible).

Les changements qui affectent une seule paire de base sur la double hélice sont appelées des mutations ponctuelles. La mutation a lieu sur un seul brin de l'ADN. C'est au moment de la réplication, et de la formation d'une nouvelle macromolécule, que cette mutation résulte en une paire de bases différente par rapport à la macromolécule originale puisque c'est la base complémentaire de la base mutée qui vient se placer sur l'autre brin en train de se créer. La mutation peut concerner soit la substitution d'une base par une autre, soit la délétion ou l'insertion d'une base (le terme de délétion plutôt que celui de suppression est employé ici par néologisme). La substitution d'une paire de bases sur l'ADN qui est ainsi observée après réplication implique bien sûr toujours une base purine appariée à une base pyrimidine (pour que la complémentarité entre les deux brins de l'ADN soit préservée). Dans le cas où la mutation

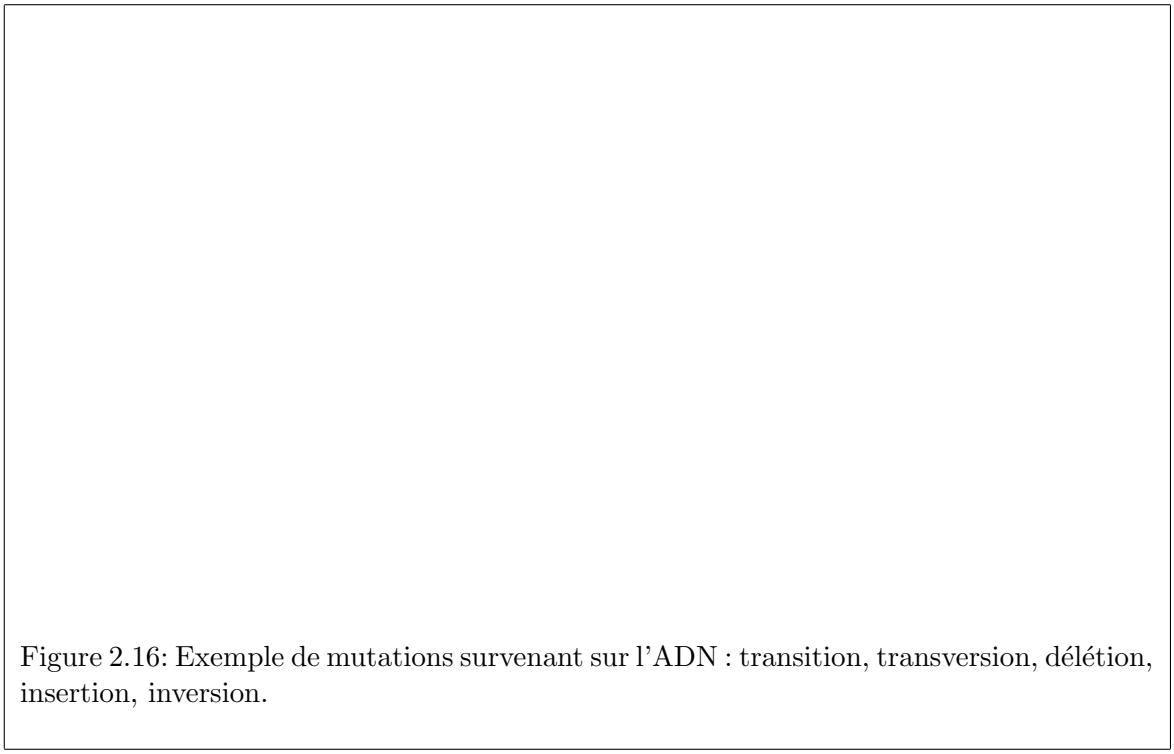


Figure 2.16: Exemple de mutations survenant sur l'ADN : transition, transversion, délétion, insertion, inversion.

initiale transforme une purine en une purine ou une pyrimidine en une pyrimidine on parle d'une transition, sinon on parle d'une transversion (voir la figure 2.16).

Des changements portant sur un nombre de paires de bases plus important dans l'ADN peuvent être rangés en deux classes principales. D'abord il y a ceux qui modifient la quantité de matériel génétique présent dans l'ADN : ce sont la délétion et la duplication de segments entiers. La taille de ces segments peut varier considérablement, et la duplication peut produire des segments identiques qui sont contigus ou distants les uns par rapport aux autres dans l'ADN. L'ordre des gènes dans le segment dupliqué peut aussi se trouver inversé par rapport à celui du segment original. Les deux autres types de changements possibles n'affectent pas la quantité de matériel génétique, mais son ordre relatif : il s'agit de l'inversion et de la translocation (aussi appelée transposition). L'inversion intervient lorsqu'un segment d'un brin de la double hélice de l'ADN est excisé puis réintégré de manière inversée ailleurs (voir la figure 2.16 — [Li and Graur, 1991]) et la translocation est un déplacement d'un segment d'ADN intra ou inter-molécules.

Ce qu'il est important d'observer dans tous les cas c'est l'effet que ces mutations peuvent avoir pour la vie de l'organisme. Cet effet est bien sûr différent suivant l'endroit où a lieu la mutation, et son ampleur n'est pas non plus toujours la même. Considérons l'exemple des mutations ponctuelles survenant dans une région de l'ADN où se situe un gène. À cause de la dégénérescence du code, la substitution d'une paire de base sur l'ADN donne ainsi lieu à un changement de codon sur l'ARN qui peut, sur le produit final qui est la protéine :

- résulter en une substitution d'un acide aminé par un autre,
- introduire un des trois codons de fin de traduction à un endroit où la traduction aurait du se poursuivre,

- coder pour le même acide aminé.

Dans ce dernier cas, on parle d'une mutation silencieuse, et de mutation neutre dans le premier cas si l'acide aminé qui se substitue à l'autre possède des propriétés physico-chimiques similaires et que la fonction de la molécule n'est ainsi pas perdue. Toute autre mutation est en général nocive et résulte en une protéine non ou partiellement fonctionnelle. Cela n'est pas toujours le cas pour les délétions ou les insertions si elles sont suivies d'autres mutations sur la même molécule d'ADN qui viennent en partie compenser l'effet de la première. Comme l'ARN est traduit en protéines à travers une relation 3:1, une seule délétion ou insertion à un endroit quelconque de l'ADN va affecter ce que l'on appelle le cadre de lecture de la traduction à partir de cet endroit. Toutefois, si une telle mutation est suivie quelques nucléotides plus loin par une opération inverse (une délétion suivie d'une insertion ou vice-versa) alors ce cadre ne sera affecté qu'entre ces deux positions et la protéine peut demeurer en partie ou totalement fonctionnelle. Le même cas peut se produire pour des délétions ou insertions survenant à la fin d'un gène ou par multiple de trois. Ces dernières mutations résultent en effet en une simple délétion ou insertion d'un (ou de plusieurs) acides aminés sur la molécule sans que les autres résidus soient modifiés.

Si les mutations surviennent dans les régions régulatrices de la transcription, ou affectent les régions régulatrices de la traduction, c'est alors l'expression des gènes qui peut se trouver modifiée. Par exemple, une translocation peut amener un gène qui se trouve dans une région non exprimée (parce que le site portant le signal de début de traduction est absent ou a été trop muté) vers une région où la transcription peut avoir lieu.

Parmi tous ces changements possibles au niveau du matériel génétique, il y en a ainsi qui ne nuisent pas à la vie de l'organisme dans lequel ils sont survenus et qui peuvent donc être préservés par transmission au cours du temps. La suite de ces changements préservés est ce que l'on appelle l'évolution. Toute une controverse existe autour de la détermination des forces qui dirigent cette évolution. La première est l'application au niveau moléculaire de la théorie néo-darwinienne de la sélection naturelle qui affirme que tout changement qui est préservé l'est parce qu'il permet à l'organisme où ce changement s'est produit, et à ses descendants à qui il le transmet, de mieux s'adapter à leur environnement et leur confère ainsi un avantage reproductif. La précision la plus importante portée à cette théorie est celle de la théorie neutraliste de Kimura [Kimura, 1983] qui affirme que la plupart des changements préservés le sont, non pas parce qu'ils confèrent nécessairement un avantage à l'organisme où ce changement a eu lieu, mais parce qu'ils ne lui sont pas immédiatement nuisibles. Si par ailleurs l'ensemble des organismes où un tel changement est survenu est à un moment donné écarté du reste de la population parmi laquelle ils vivaient, par isolement géographique ou écologique, le trait nouveau qu'a pu leur conférer le changement survenu par une mutation au hasard peut alors se trouver renforcé sans qu'il soit pour autant plus ou moins favorable à l'organisme. La théorie neutraliste n'est ainsi pas une réfutation de la sélection naturelle, mais veut modérer l'importance que celle-ci joue dans le processus évolutif au niveau moléculaire. Cette importance est aussi remise en cause quant au rôle joué par la sélection naturelle comme 'créatrice d'ordre' dans le monde vivant. La théorie proposée par Kaufmann en particulier [Kaufmann, 1993] est que la sélection naturelle ne peut pas expliquer tout l'ordre que l'on voit dans les organismes vivants, et qu'elle n'est qu'une force qui vient se greffer sur un ordre qui apparaît de manière spontanée dans tout organisme suffisamment complexe, et que c'est cet ordre-là qui en fait " *autorise, permet et limite*" l'efficacité de la sélection naturelle. Cette théorie correspond en fait à des idées exprimées par Darwin mais qui n'ont pas été reprises par ses disciples, en particulier Spencer.

Ces questions sont absolument passionnantes, mais malheureusement hors du cadre de ce travail. La chose importante à noter cependant, même si cela est exprimé de manière un peu sim-

pliste dans un premier abord, est que l'essentiel ne peut changer pour le 'bon fonctionnement' d'un organisme, c'est-à-dire que certaines choses doivent demeurer les mêmes (ou presque). La section suivante développe cette observation.

2.1.4 Conservation

Une macromolécule peut être considérée sous au moins trois aspects différents : de quoi elle est faite (sa séquence) et quelle forme elle a, à quoi elle 'sert' et quelle est son origine. Ces trois aspects sont liés, mais la relation entre eux n'est pas simple. Nous avons déjà commenté cela à propos des rapports entre séquence, structure et fonction. La relation entre ces trois éléments et l'évolution est encore plus difficile à établir. Si l'on peut affirmer par exemple de deux protéines ayant même séquence, et donc même structure et même fonction, qu'elles sont issues d'un ancêtre commun — l'expression employée dans ce cas est qu'elles sont homologues — la chose est beaucoup moins claire lorsque la similarité ne joue que sur la structure ou la fonction, ou que la similarité de séquence est faible. L'histoire est d'autant plus compliquée que si une ressemblance entre deux structures macromoléculaires peut être expliquée par le fait que celles-ci ont divergé d'une même molécule primitive, elle peut aussi être le résultat d'une convergence au niveau de la structure globale ou locale (qui se produit pour des raisons stéréochimiques).

Ce qui permet d'analyser ces macromolécules malgré tout est ce que nous avons mentionné à la fin de la section précédente : tout ne change pas. En fait, on peut même dire que beaucoup de choses changent assez peu, et surtout, que ce qui est préservé est le plus souvent lié à certaines propriétés importantes de la molécule, ADN, ARN ou protéines. Ceci est particulièrement vrai pour ce qui concerne les sites actifs et les substrats porteurs d'un signal. Il a été observé qu'à ces endroits-là, le taux de mutation est souvent bien moindre que sur le reste de la molécule — on dit que ces régions sont 'mieux conservées'. Cette conservation est donc presque toujours locale plutôt que globale, au moins par rapport aux séquences, et concerne des positions diverses sur la macromolécule qui ne sont pas nécessairement contiguës ni même proches. Comme c'est principalement la forme qui est déterminante par rapport à la fonction ou à la présence d'un signal, et que, d'autre part, plusieurs séquences peuvent acquérir la même conformation spatiale, cette conservation est également beaucoup plus fortement observée au niveau de la structure qu'à celui de la séquence [Creighton, 1993] [Chothia, 1984] [Thornton *et al.*, 1991]. Dans ce cas, elle n'est pas seulement localement mais aussi très souvent globalement vérifiée. On peut exprimer cette différence entre séquence et structure d'une autre façon en disant que la meilleure conservation de certaines régions fonctionnelles est certes relative à l'identité des monomères qui s'y trouvent, mais concerne surtout les propriétés que possèdent ces monomères et les relations qui peuvent s'établir entre eux. La structure étant un reflet direct de ces propriétés et relations, il est donc normal que sa forme soit mieux préservée.

L'étude de ces macromolécules est cependant importante aux deux niveaux de description, séquence et structure. La principale raison en est sans doute qu'aucun des deux niveaux n'apporte à lui seul toute l'information que l'on peut vouloir extraire de ces molécules (même une analyse des deux ensemble n'est souvent pas suffisante).

Celle fournie par l'analyse des structures a l'avantage d'être plus fiable dans la mesure où elle est presque palpable (on peut la 'voir'), et parce que les structures changent moins. Par ailleurs, c'est à travers les structures qu'il est possible d'envisager une étude de la dynamique des molécules qui n'est pas visible au niveau des séquences et qui peut jouer un rôle important pour la fonction que ces molécules exercent au sein des organismes. Enfin, les structures permettent une analyse plus fine des facteurs stériques et physiques contribuant à l'activité d'un site.

L'intérêt que l'on continue à porter aux séquences est quant à lui le plus souvent expliqué par le fait que l'on a pu en déterminer beaucoup plus que de structures (le rapport est environ de 20:1 et à tendance à augmenter). C'est une raison suffisante pour justifier leur analyse, mais elle n'est peut-être pas la plus essentielle. Le fait que les séquences soient non seulement moins bien conservées que les structures, mais aussi conservées localement plutôt que globalement, peut en particulier représenter à la fois une difficulté supplémentaire dans leur analyse et une source de richesse dans l'information qu'elles fournissent.

Cela peut être vrai d'au moins deux façons différentes. Considérons par exemple le cas de protéines qui possèdent la même fonction et qui ont par ailleurs la même conformation globale. Le fait justement que ces molécules soient trop bien conservées structuralement peut rendre difficile l'identification des parties nécessaires à leur activité. Si le taux de conservation au niveau des séquences est plus variable suivant la position, ce qui est presque toujours le cas, c'est une analyse de celles-ci qui peut alors permettre de localiser ces sites, soit parce qu'en général ils sont mieux conservés qu'ailleurs comme on l'a vu, soit parce qu'ils mutent de façon différente [Johnson and Overington, 1993]. Observons qu'il est important dans ce cas de travailler avec des ensembles de macromolécules dont l'homologie est due non à une duplication (paralogie) mais plutôt à un événement de spéciation (orthologie), c'est-à-dire à un événement ayant provoqué l'apparition d'espèces différentes à partir d'un ancêtre commun par suite d'une dérive géographique ou écologique d'une partie de la population. Deux macromolécules paralogues peuvent en effet avoir des fonctions et des modes de régulation divers malgré le fait qu'elles puissent par ailleurs être très proches, en particulier au niveau de leurs sites. La seconde raison pour laquelle la moindre conservation des séquences est une source potentielle d'information est que l'on peut avoir au moins autant à apprendre des différences mutationnelles mêmes subtiles pouvant survenir aux endroits importants pour le fonctionnement d'une macromolécule que de la ressemblance que ces sites peuvent présenter. Par exemple, un signal sur une molécule n'est presque jamais une simple affaire de tout ou rien. Il peut aussi être plus ou moins fort, et ainsi que l'a suggéré Lewin [Lewin, 1994], la force de ce signal peut être modulée non seulement par la présence d'autres signaux aux alentours, mais aussi justement par des types et des taux différents de mutation. Des études dans ce sens ont déjà été réalisées [Carafa *et al.*, 1990] mais il reste encore beaucoup à faire pour établir les conditions permettant éventuellement de traiter ce problème par des approches algorithmiques combinatoires.

Un des problèmes les plus fascinants consiste à établir les conditions dans lesquelles les deux approches peuvent être combinées, d'abord pour que ces approches puissent s'éclairer et s'affiner mutuellement, ensuite pour essayer, sinon de résoudre l'énigme du rapport séquence-structure, au moins de mieux comprendre comment "*l'architecture dans l'espace a pu se trouver engendrée par la simplicité d'une combinaison linéaire*" [Jacob, 1970] et pouvoir ainsi mieux apprécier toute la richesse que cette simplicité cache. Cette richesse est telle qu'apparemment, la seule loi qui semble gouverner la relation existant entre toutes les séquences produisant une même structure est celle du hasard. Mais il est difficile de distinguer le hasard d'une complexité trop grande pour qu'aucun ordre puisse y être immédiatement visible. Ce qui ne veut pas dire que cet ordre n'existe pas.

Le problème séquence-structure n'est pas directement traité dans ce travail mais tout ce qui est présenté ici tente d'aller dans cette direction, en particulier ce qui va nous concerner maintenant est la recherche d'un certain ordre dans les macromolécules, ce qui va se traduire initialement de manière plus concrète par une recherche de ressemblances. La question séquence-structure quant à elle est reprise bien plus tard.

2.2 Recherche d'un ordre

2.2.1 Motivation et critères

L'un des désirs les plus profonds de l'esprit humain est sans doute que les choses dans l'univers puissent présenter un certain ordre.

Toute tentative d'appréhension ou d'exploration de cet univers est donc principalement une recherche d'ordre et finit tôt ou tard par faire appel à l'analogie pour essayer d'attribuer une place et une fonction aux objets de l'univers. Comparer des objets, mesurer leur ressemblance, les classer, distinguer ce qui leur est essentiel de ce qui leur est purement accidentel (bien que le purement accidentel puisse aussi avoir son intérêt) sont ainsi des activités que nous sommes amenés à faire en permanence.

Nous avons vu que l'univers qui nous intéresse ici est celui de la biologie moléculaire et les objets sont les macromolécules. Déterminer ce qu'il est essentiel de caractériser sur ces objets, parfois même réussir à savoir s'il y a quelque chose à caractériser, peut être une tâche complexe. L'idée principale de l'analogie, ce qui motive l'importance qu'on lui accorde dans toute tentative de distinguer une structure dans l'univers, c'est que justement à travers elle "*l'invisible devient visible*" [Jacob, 1970], ou le devient en partie, parce qu'elle permet de repérer ce qui ne change pas, ou change peu, ou encore change d'une même façon et à un même rythme. L'hypothèse est faite que ce sont ces facteurs-là qui donnent un ordre à l'univers. La recherche d'un certain ordre va donc commencer par une recherche de facteurs qui se ressemblent, ceux qui changent peu les uns par rapport aux autres dans le temps ou dans l'espace, c'est-à-dire parmi un ensemble d'objets qui sont par ailleurs différents.

Ce premier critère étant fixé, tout n'est pas résolu pour autant car la difficulté de cette recherche va dépendre bien sûr aussi de la nature de ce qui change peu. Un objet peut être vu comme un ensemble d'objets 'concrets' plus élémentaires arrangés entre eux d'une certaine manière (les nucléotides et les acides aminés, ou à un niveau encore plus élémentaire, les atomes composant ces monomères), ou comme un ensemble de propriétés (en général physico-chimiques), ou encore comme un ensemble de relations. Il peut aussi être vu sous une forme statique ou dynamique. A priori, toute régularité observée à l'un quelconque de ces niveaux (ou à plusieurs d'entre eux simultanément) est potentiellement intéressante. La première difficulté ici consiste à établir la fréquence qu'un phénomène doit présenter pour qu'on le qualifie de régulier : est-ce une fréquence très forte ou très faible, ou simplement une fréquence différente de la normale, mais alors comment mesurer le 'normal' ? La seconde difficulté est qu'il peut y avoir un très grand nombre de types de régularités. Quelques exemples ont été mentionnés dans la première partie de ce chapitre. Si l'on ne veut pas avoir à entrer dans des considérations de valeur qui peuvent être subjectives, il faudrait pouvoir établir des définitions précises de toutes les régularités qui semblent pertinentes ou en effectuer un classement. C'est une tâche qu'il serait très intéressant d'essayer de réaliser mais qui est impossible à envisager complètement. En général, des choix sont donc faits qui dépendent de ce qui est observé, qui lui même dépend des hypothèses que les personnes se font sur le fonctionnement des objets étudiés (c'est-à-dire dépend en fin de compte de leurs expectatives). Enfin, de manière plus prosaïque, ces choix dépendent aussi de ce qu'il est possible de chercher avec les instruments dont nous disposons en un temps qui soit raisonnable.

La façon de procéder est donc celle indiquée par François Jacob [Jacob, 1970] : "*le plus simple doit éclairer le plus complexe*". C'est aussi l'idée présentée par Gombrich [Gombrich, 1987] à propos de l'exécution d'un tableau mais qui s'applique tout aussi bien à la recherche qui nous intéresse : "*À défaut d'un point de départ, d'un schéma initial, nous ne pourrions jamais parvenir à contrôler le flux mouvant de l'expérience. Nous serions incapables de classer, de mettre*

nos impressions en bon ordre, si nous ne disposions pas de catégories. Assez paradoxalement, on s'est aperçu que la nature de ces premières catégories n'avait pas une grande importance, il nous est toujours possible de l'adapter à nos besoins, ... pourvu que le dessin initial ne contienne aucune indication fausse".

Une manière de simplifier toute recherche consiste donc à imposer des limites à l'univers que l'on cherche à explorer, soit en restreignant son champ, soit en établissant des contraintes à la méthode servant à effectuer cette exploration. Les contraintes peuvent être diverses, mais l'important est qu'elles soient toujours clairement définies de telle sorte que leur introduction ne rende pas l'interprétation des résultats obtenus impossible à réaliser, soit parce qu'elles déforment la réalité, soit parce qu'elles la brouillent encore plus.

Nous allons voir dans la section suivante les principales contraintes et simplifications que nous avons choisies d'établir dans le cadre de ce travail.

2.2.2 Le cadre spécifique de notre recherche

La première contrainte que nous allons imposer au modèle adopté pour analyser et comparer les macromolécules, ADN et ARN ou protéines, est que celui-ci tienne compte de l'ordre linéaire qui existe naturellement dans ces molécules. Cela signifie que nous allons travailler avec des chaînes de caractères. Celles-ci peuvent donc représenter directement la structure primaire, c'est-à-dire la séquence, des macromolécules où, comme on l'a vu, à chaque monomère correspond un symbole sur un alphabet de taille 4 ou 20. Elles peuvent aussi, dans le cas des protéines, être obtenues de la structure par un codage des coordonnées internes de la chaîne des carbones α. Il n'est pas important pour l'instant de comprendre comment cela est fait, simplement de noter que, dans tous les cas, ce que l'on obtient est un ensemble de chaînes sur un certain alphabet plus ou moins grand, et que ce sont ces chaînes que l'on compare. Le terme générique de chaîne est adopté ici pour tenir compte du fait que les objets désignés représentent tantôt des séquences, tantôt des structures dans l'espace codées en une suite linéaire de caractères. Ce terme a été préféré à celui de texte dont nous avons voulu éviter la connotation linguistique.

La seconde contrainte va porter alors sur la façon de comparer ces chaînes. Il y en a deux principales, la première consiste à les comparer globalement, la seconde à réaliser des comparaisons locales. Dans ce second cas, il s'agit seulement de repérer certains objets qui sont communs à ces chaînes. Nous présentons ici les deux approches, globale et locale, bien que notre choix se soit porté sur la locale. Les objets en question représentent toujours pour nous des suites contiguës de symboles de l'alphabet présents dans ces chaînes. Nous utilisons le terme de mot pour désigner ces suites de symboles. Dans le cas des séquences, un mot dénote bien un motif lexical et correspond ainsi à l'usage courant du terme, dans celui des structures un mot correspond en fait à un motif structural, c'est-à-dire à une forme géométrique sur laquelle joue une contrainte de séquentialité. Ce que l'on cherche dans les deux situations ce sont des mots qui se ressemblent. La raison de ce choix est double. Le premier est que ces mots sont intéressants en eux-mêmes. L'activité d'une macromolécule étant souvent liée à certaines régions uniquement, ce sont celles-ci qu'il faut essayer d'identifier. La seconde est plus philosophique et correspond à une vision de la nature des objets comparés que nous pouvons illustrer avec l'image du bijou dans son écrin. Supposez que l'on vous présente deux boîtes avec chacune dix petites pierres semi-précieuses et que l'on vous demande d'estimer la valeur relative des boîtes. Vous n'allez sans doute pas vous préoccuper de la taille, forme ou couleur de la boîte elle-même. Celle-ci peut bien sûr avoir un effet subjectif sur votre estimation de la valeur des pierres (elles peuvent par exemple en rehausser l'éclat), mais ce sont sur ces dernières que vous allez concentrer votre attention et que vous allez donc comparer. Cette image du ou des bijoux dans leur écrin n'a

pas été choisie au hasard, elle a en effet déjà été utilisée par rapport aux macromolécules biologiques, plus spécialement les protéines [Ptitsyn and Volkenstein, 1986], l'idée étant vraiment que seulement certaines parties de la molécule sont importantes pour expliquer sa fonction (les bijoux) et que tout le reste de la structure (la boîte) ne joue qu'un rôle de protection de ces parties qui sont essentielles. Si cette hypothèse est admise, il paraît de bon sens de mesurer la ressemblance entre objets par une comparaison de ce qui leur est essentiel uniquement, pas du reste. Repérer des mots similaires d'abord est donc une façon de comparer des chaînes, cela peut en constituer en tout cas une première étape importante. Est-ce vraiment une bonne stratégie? Plus généralement, est-ce que l'image elle-même du bijou et de la boîte est correcte? Nous laissons une discussion autour de ces questions pour plus tard. Bonne ou mauvaise, cette méthode est celle que nous adoptons. Nous parlerons aussi des méthodes de comparaisons globales portant sur les chaînes entières, mais en ce qui concerne ce travail ce que nous comparons sont des mots. Notons aussi par ailleurs que, bien que nous allons nous intéresser principalement à identifier les traits communs à un ensemble de chaînes, la recherche de mots peut être faite sur une seule chaîne. Ce que nous cherchons alors sont des mots répétés et des difficultés algorithmiques supplémentaires sont introduites suivant que l'on exige que ceux-ci soient contigus ou non, ou vérifient certaines propriétés de symétrie, directes ou miroir (c'est le cas des palindromes). Comme pour les mots communs à plusieurs chaînes, les mots répétés qui peuvent nous concerner sont bien sûr non seulement ceux exactement identiques, mais aussi ceux qui sont simplement similaires entre eux.

Une troisième contrainte (en fait il s'agit d'une famille de contraintes diverses) va jouer alors précisément sur les critères permettant de mesurer le degré de ressemblance de deux ou plusieurs chaînes, ou de considérer que deux ou plusieurs mots sont similaires.

Dans le cas d'une recherche de mots, une quatrième et dernière contrainte porte sur la fréquence avec laquelle un mot ou ses similaires doit apparaître dans une ou plusieurs chaînes pour qu'on le qualifie d'intéressant. Suivant la définition de ressemblance qui est établie entre les mots, il existe des critères statistiques qui disent quelles sont les fréquences d'apparition de ces mots qui sont 'anormales' parce qu'elles sont plus grandes ou plus petites que les fréquences avec lesquelles on peut espérer trouver un mot pris au hasard dans une ou plusieurs chaînes. Dans ce travail ces critères ne sont pas utilisés directement dans la recherche des mots similaires mais plutôt a posteriori. Initialement, nous établissons simplement que les mots similaires qui vont nous intéresser sont ceux qui apparaissent au moins (ou au plus) un certain nombre de fois. Si l'intérêt que peut présenter un mot apparaissant très souvent est assez intuitif à comprendre, celle d'un mot apparaissant peu souvent peut le sembler moins et demande donc une explication. En fait cet intérêt devient plus clair dès que l'on couple à la fréquence du mot sa localisation. L'hypothèse a ainsi été faite que certains signaux indiquant la jonction exon-intron dans l'ARN messager peuvent être constitués par des suites de nucléotides qui apparaissent avec beaucoup moins de fréquence dans la région des exons proche de la jonction que dans l'intron avoisinnant.

Enfin, si nous présentons des approches variées quant à la méthode employée pour effectuer ces comparaisons une fois fixés les critères donnés ci-dessus, celle que nous avons adoptée est une approche combinatoire. Cela veut dire que nous cherchons, de manière exacte, tous les mots qui vérifient la définition de ressemblance que nous avons choisie et qui apparaissent au moins (au plus) le nombre de fois que nous avons établi dans les chaînes à comparer.

2.2.3 Un premier commentaire concernant notre approche

Avant de clore ce chapitre entièrement dédié à la biologie, nous voudrions faire un premier commentaire concernant l'approche que nous avons choisie.

La première observation porte sur le type de régularité recherché — des mots communs à un ensemble de chaînes, ou qui se trouvent répétés dans une seule chaîne. Nous ne voulons pas nous attarder pour l'instant sur les limitations que peut présenter ce choix mais il est important de noter que ce qui va nous intéresser ce sont rarement des mots pris individuellement, mais plutôt des ensembles de mots qu'unissent certaines propriétés. Par exemple, nous allons vouloir identifier tous les ensembles de mots tels que, à l'intérieur d'un même ensemble, n'importe quelle paire se ressemble suffisamment. Peu importe pour l'instant comment est définie cette 'ressemblance suffisante', ce qu'il faut observer c'est que l'objet qui nous intéresse — l'ensemble — représente souvent un niveau de description plus riche que celui des éléments qu'il contient. La caractérisation de ces ensembles va dépendre bien sûr des critères choisis pour grouper des mots, c'est-à-dire de la définition de ressemblance adoptée.

Une seconde observation concerne précisément ces définitions. Dans le cas des structures, il est relativement aisé et immédiat de déterminer les critères qui permettent de mesurer la ressemblance entre objets car ceux-ci sont des formes géométriques, et il est très intuitif de considérer deux d'entre elles comme étant similaires si elles peuvent être approximativement superposées, après éventuellement une translation et une rotation de l'une par rapport à l'autre. La définition de ressemblance est ici assez peu sujette à polémique parce que les objets en question ont une claire identité. Mais dans le cas des séquences, cela n'est plus vrai. Même à leur niveau le plus élémentaire, celui des monomères, les objets que l'on compare sont des entités à multiples facettes, composés d'un ensemble de propriétés physico-chimiques et de relations qui dépendent non seulement de l'élément concerné mais aussi des rapports qu'il entretient avec les autres éléments de la molécule et de sa position le long de la chaîne. Les critères permettant de déterminer la ressemblance éventuelle de ces objets sont donc beaucoup plus fortement liés à ce que l'on essaye d'identifier, toutes les facettes n'ayant pas la même importance suivant ce que l'on recherche. Un point de vue est ainsi introduit dès le premier abord, dont il est souvent difficile de juger la portée et les conséquences. Cela n'est pas toujours mauvais et peut même être utilisé à bon escient, mais il faut savoir que ces choix qui sont faits a priori vont se trouver nécessairement reflétés dans les résultats obtenus. Nous allons présenter dans ce travail une façon possible d'essayer de se libérer de ce point de vue.

Une troisième observation concerne le fait que, contrairement aux définitions, la recherche de mots communs est elle, par contre, complètement détachée de toute considération biologique. Comme nous l'avons vu, une fois défini un critère de ressemblance entre mots, tous les objets qui se 'ressemblent' suivant cette définition sont localisés quel que puisse être par ailleurs leur intérêt biologique. Une question que l'on peut se poser alors concerne par exemple la pertinence de l'information qu'ils représentent. Est-ce que ces mots trouvés permettent de bien discriminer soit un lien évolutif ou structural, soit une fonction ou un signal biologique éventuellement présent sur les séquences et que l'on espérait identifier par comparaison de celles-ci? Les méthodes introduites dans ce travail ne font qu'aborder cette question, qui n'est d'ailleurs pas une question à laquelle il est facile de répondre quel que soit le moyen utilisé. C'est pourtant un problème très important.

Ce que l'on peut dire pour l'instant c'est que les choix qui ont été faits ont l'avantage d'être clairs et simples. Cette simplicité justement permet finalement une assez grande souplesse dans l'exploration de la notion de ressemblance. Est-ce que cette simplicité préserve la 'véracité du monde observé' de telle sorte que le dessin initial qu'elle permet d'en fournir "*ne contienne aucune indication fausse*" pour reprendre l'observation de Gombrich [Gombrich, 1987]? C'est une interrogation sur laquelle nous reviendrons à maintes reprises au cours de ce travail. Pour l'instant, c'est de la notion de ressemblance dont nous allons parler dans le chapitre suivant.

Chapitre 3

Exploration de la notion de ressemblance

La lune? Ce n'est plus la même,
Le printemps? Ce n'est plus
Le printemps d'autrefois.
Moi seul
N'est pas changé.

Ise monogatari, 4. Ariwara no Narihira.
Anthologie de la poésie japonaise classique.
G. Renondeau (Ed. et trad.). Gallimard, 1971.

Ce chapitre réalise une exploration des diverses façons de définir et de mesurer la ressemblance d'un ensemble d'objets. Rappelons que dans notre cas, ceux-ci sont soit des chaînes de caractères sur un certain alphabet, soit des mots présents dans ces chaînes. Par ailleurs, ces chaînes représentent le plus souvent des séquences biologiques, c'est-à-dire des structures primaires de macromolécules, mais peuvent aussi être un codage linéaire de leur structure tridimensionnelle. La notion de ressemblance dont nous allons parler et les concepts que nous allons introduire pour la définir et la mesurer font ainsi référence à certains objets biologiques bien précis, mais il est important d'observer qu'ils sont généralisables à d'autres types d'objets dans d'autres domaines. En fait, tout ce que nous allons présenter ici et dans les chapitres suivants peut s'appliquer à des chaînes dénotant n'importe quel objet décomposable en unités élémentaires sur lequel un ordre de succession a pu être défini. Typiquement un texte est un tel objet, l'ordre en question porte sur les lettres qui le composent et est alors spatial : les lettres se suivent dans le texte. Mais l'ordre peut aussi être temporel et l'objet par exemple une suite d'événements qui se succèdent dans le temps.

Ce chapitre se divise en quatre parties. La première s'attache à fournir une idée intuitive de comment évaluer la ressemblance d'objets lorsqu'une contrainte d'ordre linéaire joue sur les éléments les composant. Nous introduisons ainsi le concept d'un alignement de deux chaînes permettant d'apparier un certain nombre d'éléments de ces objets. Dans le cas des macromolécules, ces éléments sont des nucléotides ou des acides aminés et nous sommes ainsi amenés à examiner les critères de ressemblance portant sur ces monomères, en tant que symboles ou en tant qu'objets 3D. Dans cette première partie, la comparaison porte toujours uniquement sur deux objets à la fois et ceux-ci sont comparés directement entre eux. Il a d'autres façons de procéder et la seconde partie est consacrée à une présentation générale de diverses méthodes de

comparaison, globale ou locale, deux à deux ou multiple, directe ou indirecte. Nous montrons en particulier tout l'intérêt, dans certains cas, d'une comparaison de plus de deux objets simultanément et ce travail porte essentiellement sur les comparaisons multiples. Dans ce but, la distinction entre méthodes directes et méthodes indirectes devient particulièrement importante (bien que des liens existent entre les deux comme nous l'indiquons). Les deux dernières parties de ce chapitre sont donc réservées à ces méthodes, les directes d'abord puis les indirectes, et à leurs façons respectives de représenter des ensembles d'objets similaires. Ces représentations constituent d'une certaine manière un résumé plus ou moins compact de l'information contenue dans les ensembles. Nous montrons enfin que dans le cas des mesures indirectes de la ressemblance, ces représentations servent à l'identification même de ces ensembles.

Observons pour conclure que la division entre mesures directe et indirecte reflète la séparation entre le travail réalisé par d'autres avant nous et le notre.

3.1 Définition de base et conventions de notation

Avant toute chose, il convient d'établir une définition plus formelle des éléments avec lesquels nous allons travailler et d'introduire certaines conventions qui sont utilisées dans tout le reste du travail.

Définition 3.1.1 *Étant donné un alphabet Σ et une chaîne s, c'est-à-dire un élément de Σ^*, nous appelons un mot u dans s un élément de Σ^k pour $k \geq 1$ tel que $s = xuy$ avec $x, y \in \Sigma^*$. Nous disons dans ce cas également que u est présent dans s. Nous désignons par $\mid u \mid = k$ la longueur de u et par λ le mot vide (par convention $\mid \lambda \mid = 0$).*

Définition 3.1.2 *Étant donné un mot u dans une chaîne s tel que $s = xuy$, la position de u dans s est égale à $\mid x \mid + 1$ (une chaîne est ainsi indexée de 1 jusqu'à $\mid s \mid$). Nous admettons que la position d'un mot dans une chaîne est totalement associée au mot et qu'un mot u dont on connaît la longueur est ainsi désigné de manière non ambiguë par cette position.*

Bien entendu, un mot u dans une chaîne s est lui-même une chaîne — notons-la s_u (voir la figure 3.1). Nous disons alors que u est une occurrence de la chaîne s_u dans une autre chaîne, ici s. Observons que la chaîne s_u n'est pas associée à une position dans une chaîne comme l'est le mot u bien que les deux objets soient identiques en tant qu'éléments de Σ^k où $k = \mid u \mid$.

Pour simplifier, et lorsqu'il n'y aura pas danger de confusion, nous confondrons parfois ces deux entités, mot u et chaîne s_u associée à u, et désignerons les deux par le terme mot.

Finalement, dans tout ce qui suit :

- $s1$,$s2$,...,sN représente N chaînes;
- $u1$,$u2$,...,uM indique M mots;
- $s_1, s_2, ..., s_n$ et $u_1, u_2, ..., u_m$ respectivement n et m symboles dans une chaîne s ou un mot u.

Les nucléotides et les acides aminés sont désignés par des lettres majuscules mais quand une définition, un exemple ou un algorithme s'applique aux deux, des lettres minuscules sont employées, éventuellement indexées. Les lettres grecques sont réservées pour le codage en coordonnées internes des structures de protéines.

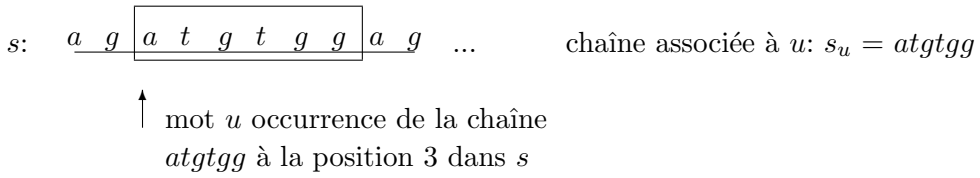


Figure 3.1: Illustration des définitions de base.

3.2 De la ressemblance des colliers de perles

3.2.1 Alignement de colliers

Une jolie image utilisée à propos des molécules biologiques est celle d'un collier de perles de couleurs différentes, les perles étant les nucléotides (4 couleurs) ou les acides aminés (20 couleurs) composant la molécule et le fil du collier les liaisons chimiques reliant un des atomes d'un monomère à un des atomes du monomère suivant le long de la chaîne nucléique ou polypeptidique.

Cette image suggère tout de suite une façon de comparer deux de ces molécules. Elle consiste à les placer allongées l'une en dessous de l'autre et à examiner chaque paire de perles mises ainsi face à face pour voir lesquelles sont identiques (ont la même couleur) et lesquelles diffèrent. Bien sûr les facteurs de variabilité étudiés au chapitre précédent font que les choses sont plus complexes. Dans le cas le plus simple correspondant aux mutations ponctuelles, certaines perles d'un collier peuvent avoir été remplacées par des perles d'autres couleurs, ou encore perdues ou rajoutées. Pour comparer les colliers, il faut donc parfois étendre le fil localement entre deux perles sur un des colliers et ne rien mettre face à une perle de l'autre. Il faut aussi considérer de déplacer un des colliers par rapport à l'autre (si ce sont les perles d'une des extrémités qui ont été égarées).

Ce processus qui consiste à placer deux objets ordonnés linéairement l'un en dessous de l'autre et à mettre certains éléments d'un objet face à certains éléments de l'autre est ce que l'on appelle réaliser un alignement. La ressemblance entre les deux objets peut alors être mesurée tout simplement en comptant le nombre de paires d'éléments mis face à face qui sont identiques, ou inversement le nombre d'éléments d'un de ces objets qui est mis face soit à un élément de l'autre ayant une nature différente soit à rien du tout. On appelle ce 'rien du tout' un trou, qui correspond ainsi pour nous exclusivement à ce qui est appelé en anglais un 'gap'. Cette terminologie est un peu trompeuse puisqu'un trou donne toujours l'idée de l'absence de quelque chose. S'il est vrai que par rapport à un alignement une insertion d'un élément dans un objet n'est qu'une délétion d'un élément dans celui qui lui fait face, en termes d'évolution les deux ne sont pas symétriques. Pour simplifier toutefois, et parce que ce que nous voyons n'est après tout que l'alignement, nous employons le terme de trou pour désigner les deux phénomènes simultanément. Il faut noter également qu'il existe en fait deux mots en anglais qui font référence à des délétions ou insertions d'éléments dans un alignement : 'indel' et 'gap'. Le premier désigne une seule délétion ou insertion tandis que le second dénote une suite contiguë de délétions. Dans l'usage courant cependant, le terme de 'gap' est utilisé dans les deux situations. Cela est le cas également pour nous en ce qui concerne le mot 'trou' qui

fait ainsi référence à une ou plusieurs délétions ou insertions dans un alignement.

Il y a bien sûr un très grand nombre de possibilités pour aligner un objet par rapport à un autre, et celle donnant lieu au plus grand nombre de paires alignées qui sont identiques (ou au plus petit nombre qui sont différentes) est alors une façon de caractériser le degré de ressemblance des deux objets. C'est la façon la plus classiquement utilisée en informatique lorsque l'on veut comparer deux chaînes de caractères puisque la distance, appelée distance d'édition, entre ces chaînes correspond au plus petit nombre d'opérations de substitutions, délétions et insertions nécessaires pour passer d'une chaîne à l'autre.

Ce n'est pas la seule manière de procéder, nous en verrons d'autres. Nous reprendrons aussi de manière plus formelle celle que nous venons de présenter. L'important pour le moment est qu'à la base de toutes les autres façons de mesurer cette ressemblance d'objets totalement ordonnés se trouve toujours, implicitement ou explicitement, un alignement de ces objets.

Une question que nous pouvons nous poser par rapport à la caractérisation de la ressemblance présentée plus haut concerne la valeur attribuée à la substitution d'un monomère par un autre : elle est en effet la même quels que soient les monomères en question. Deux d'entre eux sont ainsi toujours soit identiques, soit différents. Or nous avons vu que ces monomères sont en fait des entités complexes pouvant posséder plusieurs attributs dont certains en commun. La prochaine section est donc dédiée à une exploration de la ressemblance au niveau des perles du collier — les nucléotides et les acides aminés — et aux diverses façons d'utiliser les mesures de similarité que l'on a pu établir entre eux.

3.2.2 De la ressemblance des perles

3.2.2.1 Les nucléotides

Dans le chapitre 2, nous avons indiqué que les nucléotides sont composés de trois éléments : une base azotée, un phosphate et un sucre (ribose ou désoxyribose suivant qu'il s'agit d'ARN ou d'ADN). C'est la base azotée qui permet de différencier les nucléotides d'un même type d'acide nucléique.

Ces bases possèdent cependant certaines propriétés chimiques et structurales qui rendent possible de les grouper par classes. Nous avons mentionné un de ces groupements, composé de la classe des purines (Adénine et Guanine) et de celle des pyrimidines (Cytosine et Thymine ou Cytosine et Uracile) distinguables entre elles par le nombre de cycles que comportent les bases (respectivement 2 et 1) (voir la figure 3.2).

La capacité de chacune de ces bases à former un nombre différent de liaisons hydrogènes avec une autre permet une seconde division de celles-ci en deux nouvelles classes correspondant aux bases complémentaires Adénine-Thymine (ou Uracile) et Cytosine-Guanine, la première paire pouvant en effet établir deux liaisons et la seconde trois.

Malgré ces quelques traits en communs, il est usuel de considérer que le rôle joué par chacune de ces bases dans une chaîne nucléique lui est tout à fait spécifique et qu'une base ne ressemble ainsi qu'à elle-même. Il ne faut pas en déduire de cela cependant que les modifications pouvant survenir dans une molécule d'ADN ou d'ARN ont toutes les mêmes conséquences. Prenons le cas d'une partie de l'ADN correspondant à un gène et donc codant pour une protéine. Nous avons vu dans le chapitre précédent que la dégénérescence du code fait que certaines substitutions d'un nucléotide le long de ce gène ne changent pas le produit final, c'est-à-dire la protéine (bien que ces substitutions puissent avoir une influence plus subtile sur l'efficacité de la traduction [Lewin, 1994]). C'est ce qui a été appelé une mutation silencieuse. Une comparaison de gènes codant pour la même protéine peut ainsi amener à considérer comme semblables deux nucléotides différents mais situés à une même position relative le long du gène.

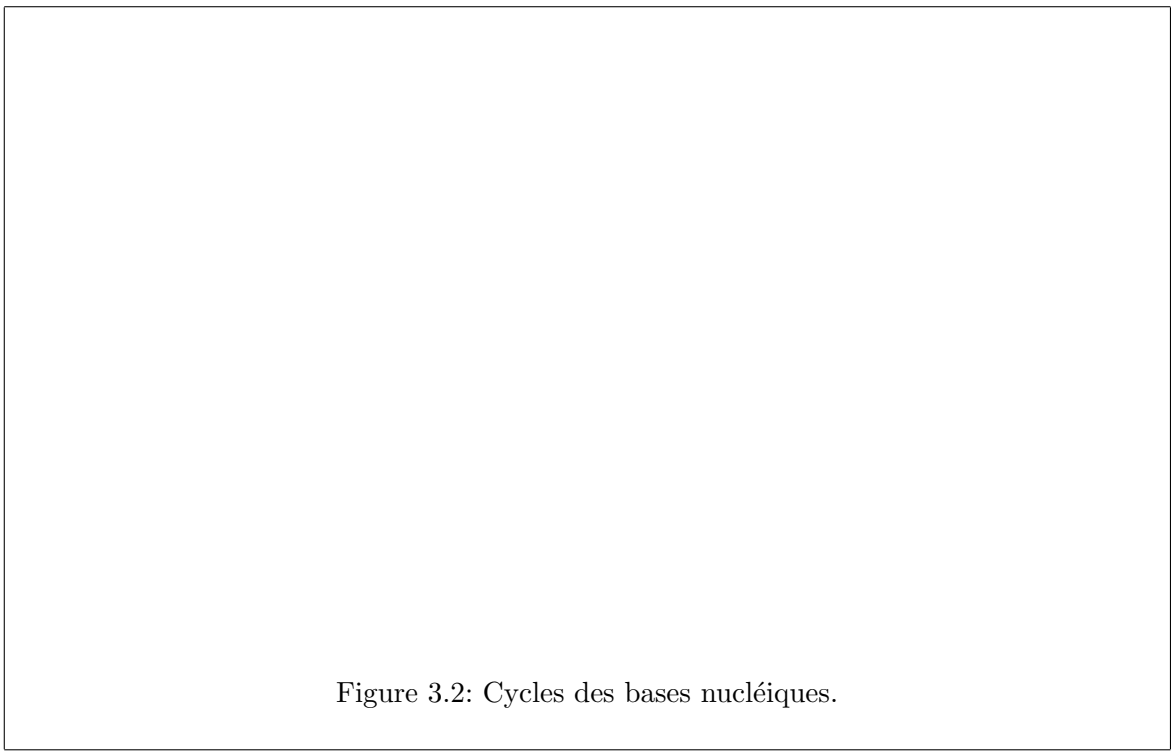


Figure 3.2: Cycles des bases nucléiques.

Une mise en correspondance d'un autre type intervient sur les nucléotides impliqués dans la formation des structures secondaires d'ARN. Ces structures sont essentiellement formées par une combinatoire de palindromes biologiques (voir la figure 2.9) et la substitution par exemple d'un nucléotide dans le brin d'une tige peut ne pas affecter la structure si elle est accompagnée d'une substitution sur l'autre brin permettant de maintenir la complémentarité des bases placées face à face (covariation). L'identification entre nucléotides est ici à la fois positionnelle et contextuelle : elle a lieu à un endroit précis et doit être accompagnée d'une autre substitution également très précise intervenant à un autre endroit structuralement lié au premier.

Enfin, une mise en correspondance plus subtile et bien plus complexe à établir met en jeu les nucléotides présents dans les régions de l'ADN ou de l'ARN impliquées dans des activités régulatrices. Ces régions sont souvent composées de suites contiguës de monomères, c'est-à-dire de mots, mais ces mots ne sont pas uniques pour un même signal (chaque mot en particulier peut dénoter une force variable de ce signal). La question est alors de savoir si l'on peut dans ce cas parler d'identification de nucléotides à certaines positions, ou si la mise en correspondance ne porte que sur le mot lui-même, et quelle est la nature de cette correspondance. En particulier, est-ce qu'elle obéit à une règle clairement définissable, dépent-elle de facteurs exclusivement internes aux signaux ou situés ailleurs? Aucune réponse définitive n'a encore été apportée à ces questions. De toute façon, il paraît assez clair que si une identification entre nucléotides est établie dans ce cas aussi, elle ne peut être que de nature positionnelle et contextuelle.

Cette étroite relation avec une position qui semble intervenir dans toute identification de nucléotides fait que l'on ne peut pas réellement parler d'une ressemblance entre deux quelconques d'entre eux en considérant uniquement leur nature chimique. En général donc, la valeur attribuée à un alignement de deux nucléotides différents est malheureusement indépendante de

leurs positions dans la chaîne.

3.2.2.2 Les acides aminés

Le tableau 2.2 présenté dans le chapitre 2 donnait une idée des propriétés physico-chimiques que possèdent les 20 acides aminés présents naturellement dans les protéines. Il montrait aussi que certaines chaînes latérales présentent des propriétés en commun. Celles-ci concernent, par exemple, leur taille, leur caractère hydrophobe ou hydrophile, ou leur charge. Tous ces facteurs sont importants pour évaluer la ressemblance de ces monomères en relation avec la structure tridimensionnelle de la protéine. Cette structure doit sa forme non pas tant à l'identité exacte des acides aminés qui la composent mais plutôt au volume qu'ils occupent dans la configuration finale et aux liens qu'ils peuvent établir avec d'autres acides aminés de la chaîne ou avec le milieu aqueux dans lequel évolue la molécule. Une similarité au niveau de quelques-unes de ces propriétés signifie ainsi que certains résidus peuvent se remplacer mutuellement sans que la structure de la protéine en soit trop affectée et par conséquent sans que soit trop modifiée son activité métabolique ou catalytique (cette activité peut éventuellement s'en trouver modulée). Au contraire des nucléotides, il est donc possible de parler d'une réelle ressemblance entre acides aminés. Le problème est de mesurer cette ressemblance.

Il y a essentiellement deux façons de le faire, reflétant deux philosophies différentes, mais produisant des résultats souvent comparables. Dans les deux cas aussi, ce qui est obtenu en dernière instance est une matrice, appelée matrice de substitution, de 20 lignes et 20 colonnes qui attribue à chaque paire de résidus une valeur numérique dont on peut dire pour l'instant de manière simplifiée qu'elle indique leur degré de similarité.

La première façon de procéder pour calculer ces valeurs consiste à réaliser une comparaison des acides aminés directement entre eux sur la base de certaines de leurs caractéristiques. Celles-ci peuvent porter tout simplement sur l'identité des acides aminés, la matrice obtenue est alors la matrice identité. Ces caractéristiques peuvent concerner également le nombre maximum de nucléotides que les codons de deux acides aminés ont en commun. L'idée est que la substitution d'un acide aminé par un autre dans une protéine se produit d'autant plus rarement que le nombre de mutations correspondantes sur l'ADN est grand. Les matrices obtenues par cette méthode de comparaison sont les matrices dites génétiques [Feng *et al.*, 1985] [Fitch, 1966] [Fitch, 1967]. Les valeurs varient donc entre 0 (les acides aminés n'ont aucun nucléotide en commun) et 3 (les acides aminés sont identiques). Enfin, les caractéristiques les plus couramment utilisées pour mesurer la ressemblance entre acides aminés portent sur leurs propriétés physico-chimiques. Les trois matrices établies de cette façon utilisent ainsi de 2 à 5 propriétés pour calculer une distance euclidienne entre les acides aminés [Grantham, 1974] [Miyata *et al.*, 1979] [Rao, 1987].

La seconde façon de mesurer la ressemblance entre acides aminés est plus indirecte et nous allons nous y attarder plus longuement. Elle consiste à examiner avec quelle fréquence les résidus se substituent les uns aux autres dans un jeu de protéines homologues, c'est-à-dire ayant un ancêtre commun. L'idée ici est que deux acides aminés se ressemblent d'autant plus qu'est grande la fréquence observée de substitution de l'un par l'autre (conservant la fonction initiale). Une telle mutation est appelée une mutation acceptée (par l'évolution). Toutes les méthodes de cette seconde catégorie commencent donc par un alignement des protéines sélectionnées. Cet alignement est établi soit d'après une comparaison directe des séquences en considérant les protéines globalement [Benner *et al.*, 1994] [Dayhoff *et al.*, 1983] [Dayhoff and Eck, 1972] [Dayhoff *et al.*, 1978] [Gonnet *et al.*, 1992] [Jones *et al.*, 1992] [McLachlan, 1971] ou localement [Henikoff and Henikoff, 1992], soit d'après les structures secondaires [Levin *et al.*, 1986], ou tertiaires [Johnson and Overington, 1993] [Miyazawa and Jernigan, 1993] [Overington *et al.*, 1992]

[Overington *et al.*, 1990] [Risler *et al.*, 1988], soit encore d'après la comparaison de plusieurs protéines dont seules les séquences sont connues avec une (ou plusieurs) protéine dont la structure aussi est connue [Luthy *et al.*, 1991]. Dans certains cas les protéines sont alignées par paires [Benner *et al.*, 1994] [Dayhoff and Eck, 1972] [Dayhoff *et al.*, 1978] [Dayhoff *et al.*, 1983] [Gonnet *et al.*, 1992] [Jones *et al.*, 1992] [Levin *et al.*, 1986] [McLachlan, 1971] [Risler *et al.*, 1988], dans d'autres l'alignement concerne un ensemble de plus de deux protéines à la fois [Henikoff and Henikoff, 1992] [Johnson and Overington, 1993] [Luthy *et al.*, 1991] [Overington *et al.*, 1992] [Overington *et al.*, 1990]. Il n'est pas essentiel pour notre propos de savoir comment précisément sont calculées les valeurs de chacune de ces matrices, mais plutôt de comprendre le principe de base de leur construction sur un exemple particulier. Nous choisissons pour cela la plus ancienne et la plus connue de toutes : la matrice de Dayhoff.

Cette matrice a été élaborée sur la base d'un certain nombre de mutations ponctuelles acceptées observées entre des protéines très proches les unes des autres (moins de 15% de résidus différents entre chaque paire). Par ailleurs, Dayhoff avait établi deux hypothèses concernant le processus d'évolution des protéines. La première était que ce processus suit un modèle Markovien de substitution d'ordre 0 des acides aminés. Ceci veut dire que l'on considère la probabilité de mutation d'un résidu à n'importe laquelle des positions de la protéine comme étant indépendante de ce qui a pu se produire dans le passé à cet endroit. Par exemple, la probabilité qu'a une cystéine d'une protéine donnée d'être transformée en une sérine est la même pour tous les résidus de cystéine de la protéine en question, quelle que soit l'identité des résidus qui ont pu occuper le site auparavant. La seconde hypothèse faite par Dayhoff était que les événements mutationnels sont indépendants du contexte, c'est-à-dire de la position et des résidus adjacents sur la chaîne.

La matrice de Dayhoff a alors été calculée à partir d'une matrice de transition dont chaque élément fournit la probabilité pour qu'un acide aminé X soit remplacé par un acide aminé Y en une unité d'évolution. Les éléments diagonaux d'une telle matrice indiquent la probabilité qu'a un acide aminé de demeurer inchangé, et leur somme la probabilité qu'il n'y ait aucune substitution dans l'intervalle d'évolution considéré. Cette matrice a été normalisée de telle façon que cette somme corresponde à une probabilité de 99%. Ainsi, l'unité d'évolution représentée par la matrice de transition correspond à une mutation ponctuelle (substitution) acceptée par 100 acides aminés (d'où le nom de PAM1). Il faut remarquer que le temps n'est pas explicitement pris en compte dans cette construction, les probabilités de substitutions sont supposées être constantes par rapport à l'unité d'évolution. Elles ne sont donc pas fonction du laps de temps nécessaire pour que cette substitution ait lieu. C'est la raison pour laquelle Dayhoff considère valable de comparer deux protéines qui ont subi des mutations sur des échelles de temps différentes, à condition que l'on puisse supposer que cette échelle est constante pour une protéine donnée. En d'autres termes, l'unité d'évolution est supposée être proportionnelle au temps avec un facteur de proportionnalité différent d'une famille à l'autre mais, pour une famille donnée, constant sur toute son histoire évolutive.

Chaque élément de cette matrice de transition indique en fait la probabilité conditionnelle pour que l'acide aminé X soit remplacé par l'acide aminé Y, étant donné que X est remplacé, multiplié par la probabilité avec laquelle il est remplacé. Cette dernière probabilité représente la 'mutabilité' (en anglais 'mutability') relative de X et est estimée par la fréquence relative du nombre de mutations subies par X. Pour chaque paire de protéines, l'exposition au changement d'un acide aminé X est la fréquence d'occurrence de X multipliée par le nombre de toutes les mutations observées à chaque 100 sites de cette paire. L'exposition totale au changement est alors la somme sur toutes les paires de ces expositions pour chaque paire. Les probabilités conditionnelles quant à elles sont évaluées sur la base des fréquences observées de substitutions entre

acides aminés des protéines sélectionnées. Celles-ci ont été choisies très proches afin d'éviter que les différences observées soient le résultat de mutations multiples, c'est-à-dire où un résidu X est remplacé par un résidu Y, qui lui-même est remplacé par un résidu Z donnant lieu à l'observation du couple XZ. De la même façon, ce choix garantit que lorsqu'aucune différence n'est observée (les acides aminés sont identiques), il n'y a pas eu de mutation (au niveau de la protéine) à cet endroit. La formule pour les éléments non diagonaux de la matrice de transition est alors la suivante :

$$\mathcal{M}(X,Y) = K.m_X.\frac{n_Y}{\sum_{X,Y} n_{XY}}$$

où n_{XY} est le nombre de substitutions observées pour la paire (X,Y), m_X est la mutabilité de X et K est une constante de pondération. La formule pour les éléments diagonaux est donnée par :

$$\mathcal{M}(X,X) = 1 - K.m_X.$$

Ainsi que nous l'avons vu, K est choisi de telle façon que $\sum_{X\in\Sigma} f_X.\mathcal{M}(X,X) = 99\%$ où f_X est la fréquence d'apparition de l'acide aminé X.

En accord avec le modèle Markovien, les matrices de probabilités correspondant à des distances évolutives plus grandes sont obtenues en multipliant la matrice PAM1 par elle-même le nombre de fois nécessaire. Ainsi, la matrice PAMn est égale à (PAM1)^n et représente la matrice des probabilités pour une séquence de protéines qui a subi n pour cent de mutations ponctuelles acceptées.

En fait, lorsque l'on veut effectuer des comparaisons entre protéines, il est plus utile d'employer une matrice dont les éléments indiquent le rapport de la probabilité qu'a un acide aminé d'être substitué par un autre à la fréquence de ces deux résidus dans des chaînes produites au hasard. Pour des raisons pratiques, les valeurs présentes dans les matrices PAM indiquent finalement les logarithmes de ces rapports (en anglais, 'log-odds'), multipliés par dix pour limiter dans les calculs en entiers les erreurs d'arrondis. Ces valeurs correspondent à des scores de substitution attribués à chaque paire de résidus. Ce score est ainsi nul, positif ou négatif suivant que les fréquences d'appariements observés dans les protéines alignées ayant même origine et exerçant une même activité sont égales, supérieures ou inférieures aux fréquences que le hasard laisserait prévoir.

Bien que parfois différentes dans leur approche, en particulier concernant les hypothèses faites sur le processus d'évolution des protéines, le mode de construction des autres matrices apparentées à celle de Dayhoff repose sur les mêmes principes de base et leurs scores indiquent ainsi toujours le logarithme du rapport pondéré de la probabilité qu'il y ait substitution d'un certain acide aminé X par un autre Y dans des protéines homologues (en anglais 'target frequency') par la probabilité que cette substitution se produise au hasard ('background frequency') qui est donnée dans l'hypothèse d'indépendance par le produit des fréquences d'apparition de X et Y [Altschul, 1991] [Karlin and Altschul, 1990] :

$$score(X,Y) = (\ln \frac{q_{XY}}{f_X.f_Y})/\lambda$$

où q_{XY} est la 'target frequency', f_X, f_Y sont les fréquences d'apparition de X et Y respectivement et λ est un facteur de normalisation.

La première question que l'on peut se poser quant à la fiabilité de toutes ces matrices concerne le fait que les protéines du jeu sélectionné doivent se trouver préalablement alignées. Il faut ainsi avoir une certaine confiance dans l'alignement initialement établi puisque de sa qualité va dépendre celle des scores de substitutions obtenus à la fin. Or la bonne réalisation d'un tel alignement implique déjà une certaine connaissance du poids à attribuer à l'appariement des

acides aminés. Par ailleurs, les mutations observées entre séquences homologues ne mettent pas seulement en jeu les substitutions mais aussi des délétions et insertions. Il faut donc pouvoir placer correctement les trous dans l'alignement. Il y a eu diverses façons d'aborder ces problèmes. La plus simple, employée par Dayhoff, consiste ainsi que nous venons de le voir à travailler avec des séquences très proches en termes d'évolution. Dans ces circonstances, peu de mutations sont survenues (en particulier très peu de délétions et insertions) et il est relativement aisé d'obtenir un alignement exact par minimisation du nombre de ces mutations observées entre chaque séquence et un ancêtre commun présumé. Une autre façon de procéder consiste à compter les substitutions dans les segments alignés sans trous qui correspondent aux régions les mieux conservées du jeu de protéines sélectionnées. C'est la méthode de Henikoff, qui de plus aligne plusieurs séquences simultanément et non plus deux seulement à chaque fois. Nous verrons plus tard pourquoi cela peut produire un alignement plus sûr. Une troisième façon de faire consiste à aligner des structures et non des séquences ce qui rend l'équivalence entre résidus plus facile à établir avec certitude. Plusieurs méthodes s'appuient ainsi, lorsque cela est possible, sur des comparaisons structurales pour établir leurs scores de substitutions. Enfin, quelle que soit la façon d'établir l'alignement initial, les erreurs que celui-ci présente éventuellement peuvent être rectifiées par une application itérative de la méthode servant à obtenir les valeurs de la matrice. Celle-ci peut en effet être utilisée pour produire un second alignement des séquences de départ, donnant lieu à une seconde matrice de substitution, qui est employée pour réaliser un troisième alignement et ainsi de suite jusqu'à convergence des valeurs de la matrice. Cette méthode de raffinements successifs est celle adoptée par Henikoff, Jones, Gonnet et Benner. Bien sûr elle ne peut garantir absolument la convergence sur des valeurs correctes si l'alignement initial est violemment faux.

On peut aussi s'interroger sur la sélection même du jeu initial. Toute mesure établie sur la base de fréquences observées reflète d'autant mieux la réalité du phénomène étudié que le nombre d'observations effectuées est grand. Dans le cas présent, il faut donc pouvoir disposer de nombreuses protéines. Le jeu avec lequel Dayhoff a initialement travaillé comportait 71 molécules. Gonnet et Benner [Benner *et al.*, 1994] [Gonnet *et al.*, 1992] ont refait des calculs très similaires à ceux de Dayhoff mais sur toute une banque de séquences cette fois (ce sont ainsi 1,7 millions de paires de séquences de la version 64 de la base MIPS [Gonnet *et al.*, 1992] qui ont été alignées et analysées). En particulier, ces séquences se trouvaient à toutes sortes de distances évolutives les unes par rapport aux autres et les matrices correspondant à des distances inférieures à 100 PAM ont été déterminées directement, sans extrapolation, contrairement aux matrices de Dayhoff. Les critères utilisés pour garantir une certaine fiabilité aux alignements nécessaires pour calculer des matrices à ces distances étaient essentiellement que les séquences impliquées devaient être suffisamment longues (comportant plus de 100 acides aminés) et le score final de l'alignement (obtenu par une technique de raffinements successifs) supérieur à un certain seuil. La valeur de ces critères a elle-même été établie par simulation sur des paires de séquences artificiellement engendrées à des distances PAM spécifiques l'une de l'autre.

Les résultats obtenus sont intéressants dans la mesure où ils montrent qu'il n'y aurait pas une simple différence dans la quantité de mutations observées entre séquences proches et séquences éloignées en termes d'évolution, mais aussi une différence dans le type de mutations produites. Ainsi pour de faibles distances évolutives, les valeurs de la matrice de substitution reflèteraient les proximités entre les codons des acides aminés, tandis que pour de grandes distances ces valeurs révèleraient plutôt les similarités physico-chimiques entre les acides aminés. Les travaux de Gonnet et Benner, ainsi que ceux de Wilbur [Wilbur, 1985] auparavant, semblent indiquer que l'hypothèse d'un modèle Markovien de substitution entre acides aminés est incorrecte (Wilbur argumente en outre que les fréquences de substitutions observées entre acides aminés

qui sont la conséquence du remplacement de deux ou trois nucléotides sont plus grandes que celles que l'on pourrait déduire des fréquences de substitutions n'exigeant le remplacement que d'un seul nucléotide). Une matrice obtenue par comparaison de séquences très proches comme cela est le cas de Dayhoff ne serait en tout cas pas une bonne base de départ pour l'obtention de matrices pouvant servir à comparer des séquences plus distantes. Ces matrices sont sans doute construites de manière plus fiable à partir de l'observation directe de séquences évolutivement éloignées comme le font Gonnet et Benner, mais également Henikoff, Overington, Johnson et Lüthy. Dans le cas de Henikoff, plusieurs matrices sont calculées comme pour la série PAM, chacune appropriée à une certaine distance d'évolution (la matrice BLOSUM n correspond ainsi à une distance où au moins n% d'identité est observé entre séquences).

Un troisième problème qui surgit alors est que cette distance n'est bien sûr pas connue à l'avance pour les séquences que l'on veut comparer. Une façon d'éviter la difficulté consiste à utiliser plusieurs matrices à la suite, chacune correspondant à une distance différente [Altschul, 1993] [Collins *et al.*, 1988] et de choisir celle fournissant le 'meilleur résultat', en général celle rapprochant le plus les deux séquences comparées dans un sens qui sera rendu précis plus loin. Une autre hypothèse est cependant faite ici, concernant cette fois l'homogénéité des séquences par rapport au temps de l'évolution. Lorsque plusieurs séquences sont en jeu, comme cela va être le cas pour nous, celles-ci peuvent former divers groupes distincts en termes de distance évolutive. Aucune matrice ne peut alors être la plus appropriée simultanéement pour tous ces groupes.

Par ailleurs, nous avons vu dans le cas des nucléotides que toute comparaison de ces éléments est indissociable de leur contexte dans une séquence. L'idée intuitive que l'on se fait des acides aminés est qu'il est possible de dire quelque chose à propos de la ressemblance de deux d'entre eux dans l'absolu. D'une certaine façon cela est vrai, il est possible de mesurer une ressemblance entre eux soit directement, soit indirectement à travers des alignements comme nous venons de le montrer, et il est sans doute très significatif que les matrices qui ont été établies par cette dernière méthode reflètent souvent les propriétés physico-chimiques de ces résidus même lorsque ceux-ci n'ont pas du tout été pris en considération lors de leur élaboration. Cela est le cas en particulier de la matrice PAM250 de Dayhoff ainsi que le montre la figure 3.3. Mais les mesures fournies par ces matrices représentent seulement une moyenne de diverses caractéristiques. Or un acide aminé n'est qu'un élément dans une molécule et le rôle qu'il joue dans la structuration de celle-ci ou dans la fonction qu'elle exerce a une importance variable suivant la position qu'il y occupe et peut ne concerner à chaque fois qu'une partie de ses propriétés. La ressemblance entre acides aminés est donc elle aussi fortement dépendante du contexte. Ainsi, par exemple, l'arginine qui est normalement rangée parmi les acides aminés polaires (voir la figure 3.11 plus loin) peut présenter un comportement hydrophobe lorsque localisée dans une hélice α [Cornette *et al.*, 1987]. Des travaux récents, notamment de la part d'Overington [Overington *et al.*, 1992] [Overington *et al.*, 1990] et de Lüthy [Luthy *et al.*, 1991], ont porté sur l'observation des taux de substitution des résidus en fonction de ce contexte. Les valeurs numériques obtenues, et donc les scores, dépendent ainsi de l'emplacement de ces résidus dans la molécule et, en particulier, du type de structure secondaire dans laquelle ils se situent. Ces auteurs fournissent non pas une mais à nouveau plusieurs matrices, chacune donnant cette fois les scores des substitutions observées entre acides aminés faisant partie par exemple soit d'une hélice α, soit d'un feuillet β, soit encore d'un coude (en anglais 'turn'). Ces matrices 'contextuelles' sont très intéressantes mais posent une question importante : comment en effet s'en servir lorsque l'on veut comparer des macromolécules dont seules les séquences sont connues?

Une autre approche du problème du contexte est celle adoptée par Gonnet [Gonnet *et al.*, 1994] qui a étudié les fréquences des substitutions observées pour les dipeptides cette fois,

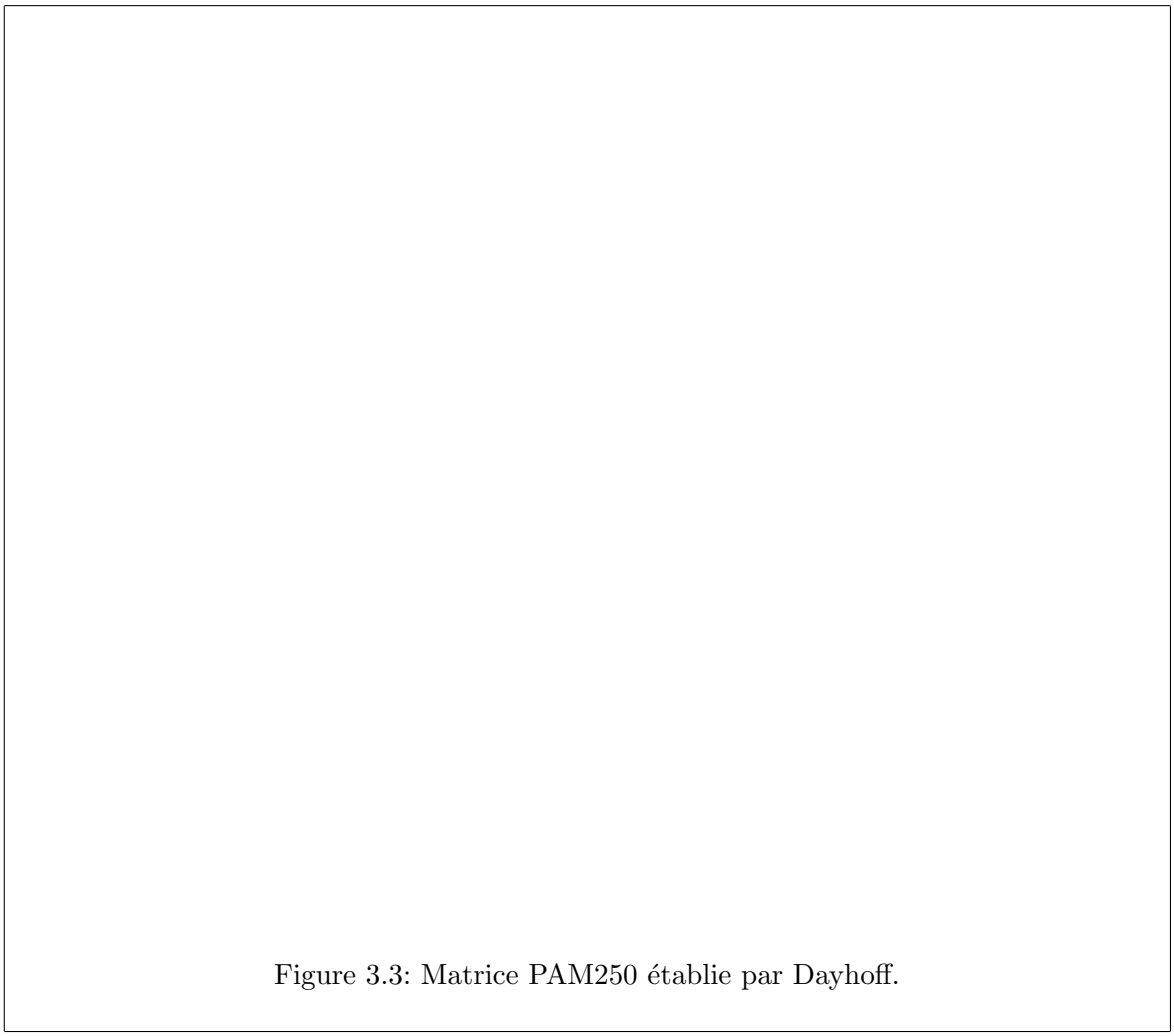


Figure 3.3: Matrice PAM250 établie par Dayhoff.

où un dipeptide est une suite contiguë de 2 acides aminés. Les matrices obtenues ont ainsi 400 lignes et 400 colonnes. Gonnet montre que les fréquences trouvées sont souvent différentes du produit des fréquences des substitutions observées pour les acides aminés, rendant fausse l'autre hypothèse importante du modèle employé par Dayhoff et qui concernait l'indépendance des positions. Gonnet n'a cependant examiné que les interactions au voisinage immédiat d'un résidu, alors que d'autres plus lointaines existent sans doute aussi. Par ailleurs, même en ce qui concerne ces interactions très proches, on ne dispose pas d'assez de données pour que les matrices établies soient significativement éloignées de celles couramment utilisées.

Pour l'instant, il faut noter que, quelles que soient les limitations des valeurs fournies par ces matrices de substitutions, en pratique elles fournissent une bonne évaluation de la ressemblance entre acides aminés. Cela est à la fois positif et négatif. C'est positif car cela signifie que le choix d'une matrice est, dans une bonne partie des cas, moins crucial que l'on ne pourrait le craindre. Les matrices les plus connues et les mieux établies sont alors plus ou moins équivalentes dans leur pouvoir de mesurer cette ressemblance [Risler *et al.*, 1988] [Vogt *et al.*, 1995] (voir la figure 3.4) : il s'agit en particulier de la série PAM des matrices de Dayhoff, ou de ses dérivées [Benner *et al.*, 1994] [Gonnet *et al.*, 1992] [Jones *et al.*, 1992], et de la série BLOSUM des matrices de Henikoff. Cette bonne performance en pratique est négative dans le sens où elle est justement un reflet de cet effet 'moyenne' des mesures obtenues. Elles 'marchent bien' en général mais sont-elles capables d'évaluer correctement les ressemblances plus difficiles à distinguer, plus subtiles? Les éléments présentés dans ce chapitre et dans le précédent montrent ainsi que certains attributs des acides aminés, et cela vaut aussi pour les nucléotides, sont plus importants que d'autres pour mesurer leur similarité et que cette importance varie en particulier suivant l'endroit où se trouvent ces monomères et la fonction qu'ils exercent.

3.2.2.3 Les perles comme des objets 3D

Nous avons vu dans le chapitre précédent que l'activité exercée par les macromolécules dans l'organisme peut dépendre fortement de la forme géométrique que ces objets adoptent dans l'espace. Or la structure tridimensionnelle d'une macromolécule est déterminée par l'arrangement spatial des atomes des monomères la composant, et ceux-ci forment une chaîne. Une représentation de la structure qui préserve cet ordre linéaire nous permet ainsi de les considérer comme des chaînes de caractères. Avant de mesurer la ressemblance des monomères en tant qu'objets 3D, il nous faut voir comment les coder en symboles d'un alphabet géométrique.

Considérons par exemple le cas d'une protéine, et voyons comment établir le codage de sa structure en une chaîne. Tous les atomes n'ont pas besoin d'être pris en considération pour cette représentation en chaîne et il est possible de ne considérer en fait, pour chaque résidu, que son carbone α, l'atome d'azote du groupement amino NH_2 et l'atome de carbone du groupement carboxyl COOH. La chaîne de ces atomes pour tous les résidus constitue ce qui est appelé le squelette de la protéine. La conformation locale de ce squelette peut être définie à chaque résidu par trois coordonnées internes habituellement appelées angles dihédraux : Φ, Ψ and ω [Creighton, 1993] [Ramachandran *et al.*, 1963]. Φ et Ψ correspondent aux angles qu'établissent entre eux, d'un côté le plan où se situe la liaison peptidique, et d'un autre celui contenant soit l'atome de carbone soit celui d'azote respectivement, ainsi que l'atome de carbone α le plus proche et la chaîne latérale reliée à cet atome (voir la figure 3.5). L'angle ω quant à lui correspond à un angle de torsion autour de la liaison peptidique. Pour des raisons chimiques, ω est fixé (à 0 ou 180°) et peut être ignoré en première approximation, de sorte que les coordonnées internes du squelette sont constituées des seuls angles Φ et Ψ que l'on représente sur une carte planaire Φ, Ψ nommée carte de Ramachandran [Ramachandran *et al.*, 1963]. La structure du

Figure 3.4: Une comparaison par analyse factorielle de diverses matrices de substitution réalisée par Risler *et al.* [Risler *et al.*, 1988] : DA (Dayhoff), FE (Feng), GR (Grantham), LE (Levin), MI (Miyata), ML (McLachlan), RI (Risler).

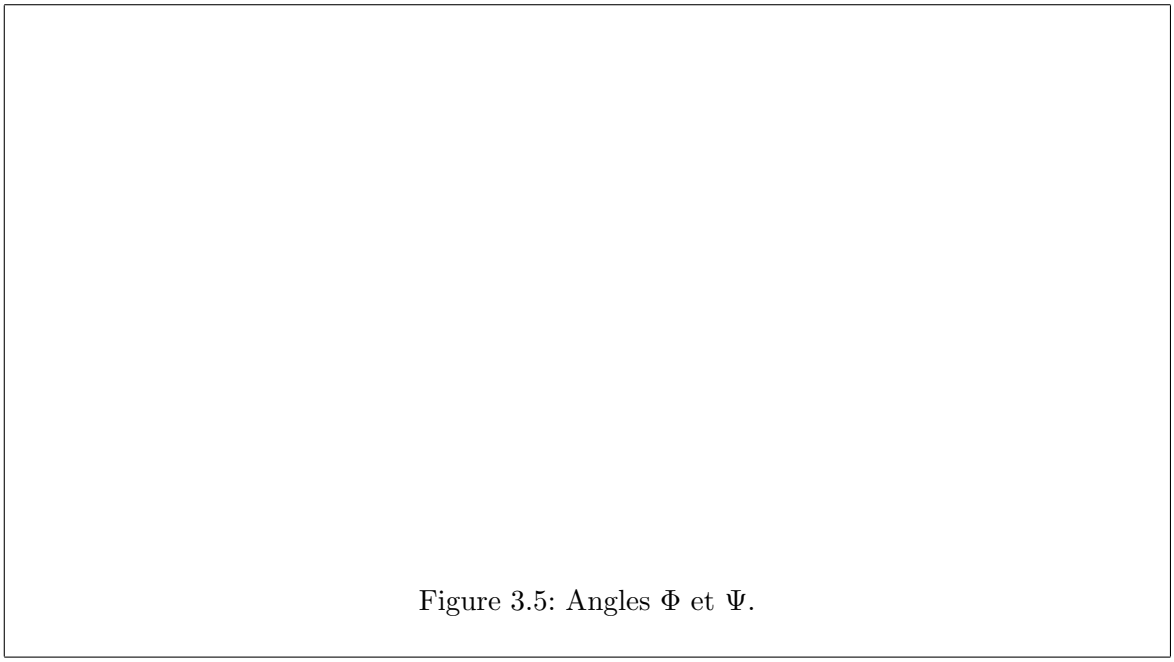
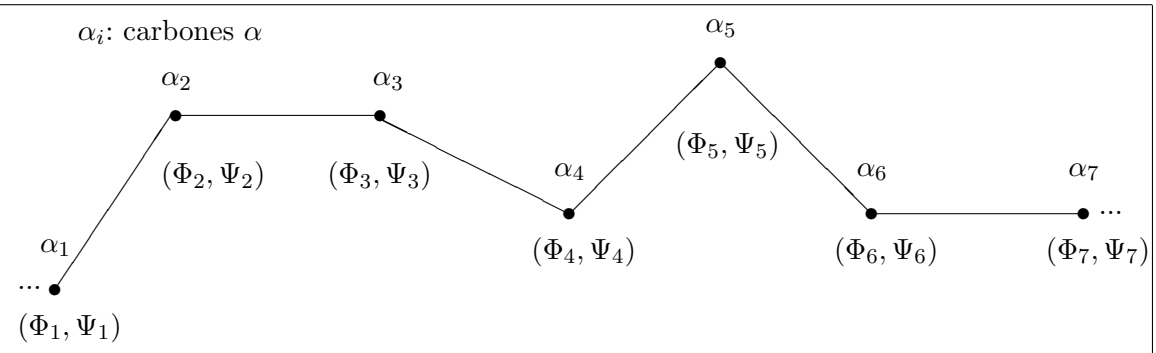


Figure 3.5: Angles Φ et Ψ.

squelette peut ainsi être définie de manière unique par la succession linéaire des paires d'angles (Φ, Ψ) le long de ce squelette. Il est habituel de ne représenter que les carbones α de ce dernier comme sur la figure 3.6. Observons qu'alors les traits qui sont dessinés entre deux de ces carbones ne sont que des pseudo-liens qui ne correspondent pas à des liaisons chimiques. Ces traits ont des longueurs approximativement égales car les distances entre carbones α d'une chaîne polypeptidique sont à peu près constantes (environ 3,8 Angström). Comme ces paires d'angles (Φ, Ψ) représentent des paires de valeurs réelles, elles doivent être recodées sur des valeurs discrètes de manière à pouvoir travailler avec un alphabet de symboles discrets. Pour réaliser cela, une grille de maille ϵ est construite sur la carte de Ramachandran [Pothier, 1993] [Sagot *et al.*, 1995b] (voir la figure 3.10 plus loin — notons aussi que ce qui apparaît comme étant plat sur la figure représente en fait la surface d'une sphère). Le centre de chaque carré de la grille — on appelle ce centre un noeud — reçoit un numéro allant de 1 à $(360/\epsilon)^2$. Ensuite, toute paire d'angles (Φ, Ψ) est mise en relation avec le centre du carré à l'intérieur duquel elle se place. Par conséquent, à chaque paire (Φ, Ψ) de la protéine correspond à la fois un noeud et le numéro qui a été attribué à celui-ci, mais il est clair qu'à un noeud donné, et donc à un numéro, peuvent correspondre plusieurs paires d'angles identiques ou très proches les unes des autres (toutes se référant à des points internes à une même maille). Une paire d'angles quelconque se trouve ainsi codée sur un seul symbole.

La ressemblance entre monomères comme des objets dans l'espace est, dans ce cas précis, immédiate à établir : elle va porter sur une ressemblance entre paires d'angles et donc, après codage, à une relation de voisinage sur les carrés de la grille ainsi que nous le montrons plus loin. Observons que, si l'ordre linéaire des monomères n'était pas pris en considération, c'est-à-dire, si nous voulions les considérer non plus comme des résidus le long d'une chaîne mais comme des objets libres dans l'espace, il ne serait plus possible d'attribuer à chacun un code qui permette de comparer deux monomères en tant qu'entités indépendantes. Tout codage dans ce cas dépendrait de l'existence d'un système de référence, qui pourrait être local ou global, mais qui ne pourrait être absolu.



Structure d'une protéine codée initialement comme :
$$...(\Phi_1, \Psi_1)(\Phi_2, \Psi_2)(\Phi_3, \Psi_3)(\Phi_4, \Psi_4)(\Phi_5, \Psi_5)(\Phi_6, \Psi_6)(\Phi_7, \Psi_7)...$$

Figure 3.6: Structure du squelette d'une protéine définie comme une succession linéaire de paires d'angles (Φ, Ψ).

3.2.3 Matrice, relation, couverture

3.2.3.1 Les matrices

Rappelons que notre but dans cette discussion sur la ressemblance des perles était celui d'établir le poids à donner à l'appariement de deux monomères dans un alignement. Une manière de procéder consiste à faire appel à une matrice comme celles vues précédemment. La façon la plus immédiate d'utiliser les mesures de ressemblance présentes dans une de ces matrices est d'attribuer directement leurs valeurs à cet appariement. Observons que dans le cas des nucléotides, cette matrice demeure le plus souvent l'identité.

Ces matrices en fait ne fournissent pas le même type d'information et ne possèdent pas toutes les mêmes propriétés mathématiques. Dans le cas des protéines, celles obtenues par comptage des mutations observées sont, à l'exception de celle de Risler, toutes des matrices de proximité. Celle de Risler ainsi que les matrices de Grantham, de Miyata et de Rao sont des matrices de distances. Finalement les matrices génétiques sont des matrices de similarité.

Commençons par établir certaines définitions [Cailliez and Pages, 1976] :

Définition 3.2.1 *Un indice de proximité sur un ensemble E est une application f de $E \times E$ dans l'ensemble $\mathcal{R}$ telle que :*

1. *$f(i,j) = f(j,i)$ pour tout $(i,j) \in E \times E$;*
2. *$f(i,j) =$ est d'autant plus élevé que i est proche de j, pour tout $(i,j) \in E \times E$.*

Définition 3.2.2 *Un indice de similarité sur un ensemble E est une application f de $E \times E$ dans l'ensemble $\mathcal{R}$ telle que :*

1. *$f(i,j) = f(j,i)$ pour tout $(i,j) \in E \times E$;*
2. *$f(i,i) =$ maximum parmi toutes les valeurs attribuées aux éléments de $E \times E$, cela pour tout $i \in E$.*

Définition 3.2.3 *Un indice de dissimilarité sur un ensemble E est une application f de $E \times E$ dans l'ensemble $\mathcal{R}^+$ des nombres positifs ou nuls telle que :*

1. $f(i,j) = f(j,i)$ *pour tout* $(i,j) \in E \times E$;
2. $f(i,i) = 0$ *pour tout* $i \in E$.

Définition 3.2.4 *Une distance sur un ensemble E est un indice de dissimilarité sur E qui vérifie les deux propriétés supplémentaires suivantes :*

1. $f(i,j) = 0 \Longrightarrow i = j$;
2. $f(i,k) \leq f(i,j) + f(j,k)$ *pour tout* $i,j,k \in E$ *(inégalité triangulaire).*

Utilisant ces divers indices, nous pouvons alors établir des définitions formelles pour les matrices construites dans la section précédente.

Définition 3.2.5 *Soit Σ un alphabet. Une matrice de proximité/similarité est un indice de proximité/similarité de $\Sigma \times \Sigma$ dans $\mathcal{R}$.*

Définition 3.2.6 *Une matrice de proximité/similarité $\mathcal{M}$ normalisée est une matrice telle que, pour tout $a, b \in \Sigma$:*

1. $0 \leq \mathcal{M}(a,b) \leq 100$.

Définition 3.2.7 *Soit Σ un alphabet. Une matrice de dissimilarité/distance est un indice de dissimilarité/distance de $\Sigma \times \Sigma$ dans $\mathcal{R}^+$.*

Malgré leurs différences, des liens existent entre certaines de ces matrices. Ainsi, une matrice de distance peut facilement être transformée en une matrice de similarité. Il suffit de la normaliser, puis d'attribuer à chaque paire d'éléments de l'alphabet (nucléotides ou acides aminés) un score égal à 100, moins le score que la paire possédait dans la matrice d'origine. L'opération inverse, à savoir la même démarche effectuée sur une matrice de similarité, ne nous fournit pas nécessairement une matrice de distance, la raison étant que l'inégalité triangulaire n'y est pas toujours observée.

La non obéissance par les matrices de proximité, comme par exemple les matrices PAM et BLOSUM, de la règle 2 relative aux indices de similarité peut quant à elle paraître non intuitive. Elle s'explique facilement si l'on se souvient que les valeurs attribuées aux éléments de la diagonale principale de ces matrices indiquent la probabilité de retrouver le même acide aminé à une même position après n mutations. Il n'y a pas de raison pour que cette probabilité soit la même pour tous les résidus. Ceux en particulier possédant un ensemble de propriétés peu communes mutent rarement entre protéines homologues, un remplacement de l'un d'entre eux à un endroit quelconque de la molécule risquant de transformer complètement la structure de celle-ci. Cela est le cas de la cystéine par exemple qui est un résidu possédant plusieurs fonctions uniques comme celle de lier les atomes métalliques et d'établir des ponts disulfures.

Dans la plupart des cas cependant, lorsque nous utilisons ces matrices pour attribuer un score à l'appariement de deux monomères dans un alignement, nous travaillons avec des matrices de similarité normalisées entre 0 et 100. Une telle matrice, dérivée de la matrice PAM250 de la figure 3.3 et corrigée par Brutlag [Brutlag *et al.*, 1990] pour tenir compte de certaines des critiques de Wilbur, est indiquée dans la figure 3.7.

	S	G	A	I	V	L	T	D	E	R	K	M	F	Y	C	P	N	Q	H	W
S	17	17	17	6	8	6	15	10	10	8	12	4	4	3	8	13	11	8	6	3
G	17	48	20	5	9	5	12	13	12	5	9	4	3	2	4	11	12	8	5	1
A	17	20	23	9	13	9	17	12	12	7	11	6	4	3	5	16	11	9	7	1
I	6	5	9	18	20	18	8	3	3	3	6	7	8	3	3	4	3	3	2	1
V	8	9	13	20	30	20	11	5	5	4	7	10	6	3	5	7	5	5	4	1
L	6	5	9	18	20	60	8	4	4	5	7	20	16	7	1	6	5	6	5	4
T	15	12	17	8	11	8	18	8	8	6	12	5	3	2	4	10	9	6	5	2
D	10	13	12	3	5	4	8	20	18	5	11	2	1	1	1	6	12	10	7	1
E	10	12	12	3	5	4	8	18	21	5	11	2	1	1	1	7	10	13	7	1
R	8	5	7	3	4	5	6	5	5	29	23	3	2	1	2	7	6	8	8	6
K	12	9	11	6	7	7	12	11	11	23	42	9	2	2	2	8	13	12	9	3
M	4	4	6	7	10	20	5	2	2	3	9	11	4	1	1	3	2	3	1	1
F	4	3	4	8	6	16	3	1	1	2	2	4	58	31	1	2	2	1	3	4
Y	3	2	3	3	3	7	2	1	1	1	2	1	31	55	5	1	3	1	4	3
C	8	4	5	3	5	1	4	1	1	2	2	1	1	5	93	3	2	1	2	1
P	13	11	16	4	7	6	10	6	7	7	8	3	2	1	3	35	7	8	6	1
N	11	12	11	3	5	5	9	12	10	6	13	2	2	3	2	7	11	7	8	1
Q	8	8	9	3	5	6	6	10	13	8	12	3	1	1	1	8	7	16	11	1
H	6	5	7	2	4	5	5	7	7	8	9	1	3	4	2	6	8	11	26	1
W	3	1	1	1	1	4	2	1	1	6	3	1	4	3	1	1	1	1	1	100

Figure 3.7: Matrice PAM250 obtenue à partir de la matrice de la figure 3.3, corrigée par Brutlag et normalisée entre 0 et 100.

Nous allons voir plus loin que l'emploi direct de ces matrices conduit à l'affectation d'une valeur à un alignement qui est elle-même une mesure moyenne de la ressemblance des molécules comparées, reflétant ainsi la caractéristique principale de la construction de ces matrices. Mais celles-ci peuvent également servir de base de départ à d'autres façons de mesurer la ressemblance qui sont particulièrement appropriées à la réalisation de comparaisons multiples. C'est ce que nous allons voir maintenant.

3.2.3.2 Relation

3.2.3.2.1 Relation sur l'alphabet des monomères

La première approche correspond à une discrétisation de la notion de ressemblance. Cela paraît être un retour en arrière en quelque sorte puisque deux éléments sont à nouveau considérés comme étant soit similaires, soit non similaires, sans moyen terme entre les deux. Mais cette fois-ci, la similarité porte sur plus que la simple identité des monomères en jeu. Nous dirons ainsi que deux monomères se ressemblent s'il existe une relation entre eux établie sur la base de quelques-unes de leurs caractéristiques. La motivation principale de cette approche est qu'elle permet une première formalisation de la relation entre plus de deux objets à la fois qui sera mathématiquement bien établie. Mais commençons par voir comment définir une telle relation.

La manière la plus classique consiste à établir un seuil sur une des matrices de substitution vues précédemment [Brutlag *et al.*, 1990] [Cobbs, 1994] [Soldano *et al.*, 1995]. Nous obtenons alors :

Définition 3.2.8 *Soit Σ un alphabet et $\mathcal{M}$ une matrice de similarité (ou de proximité). Une relation R_{seuil} de similarité sur $\Sigma \times \Sigma$ est définie par :*

$$\textit{pour tout } a,\ b \in \Sigma,\ a\ R_{seuil}\ b$$
$$\Updownarrow$$

$$\mathcal{M}(a,b) \geq \textit{seuil}$$
$$\underline{\textit{ou}}$$
$$a = b \textit{ (a et b désignent en fait le même monomère)}$$

où 'seuil' est une valeur prise entre les valeurs minimum et maximum de $\mathcal{M}$.

Une définition semblable peut être établie pour une relation obtenue à partir d'une matrice de distance. Bien entendu, si la matrice est l'identité (c'est le cas pour l'ADN et l'ARN), la relation obtenue est alors l'identité.

Notation 3.2.1 *On note I la relation d'identité.*

Il est clair que la relation R_{seuil} est symétrique (puisque la matrice l'est) et réflexive (par construction) mais qu'en général cependant elle n'est pas transitive. Elle n'est donc le plus souvent pas une relation d'équivalence et ne définit donc pas toujours une partition de l'ensemble des monomères. Une manière à la fois d'illustrer ce phénomène de non-transitivité et surtout de sentir le caractère subjectif qui peut intervenir quant au choix du seuil permettant d'établir une telle relation est donné dans la figure 3.8. Il est certain que la ressemblance du dessin a avec le dessin b est frappante. Il en est de même pour celle du dessin b avec c, et de c avec d. Mais où placer le seuil qui permet de distinguer ces objets? Ceci n'est qu'une image, mais il est évident que le même problème se pose ici aussi, et nous observons ainsi l'introduction d'un nouveau point de vue dans la notion de similarité entre monomères, outre celui déjà introduit lors de la construction de la matrice.

Bien entendu, la relation de similarité entre monomères peut être établie directement, sans utilisation d'une matrice. Nous employons alors la notation simplifiée R. Ce qui est intéressant pour nous, c'est que quelle que soit la façon de définir une telle relation R, celle-ci conduit de manière immédiate aux notions de clique et de clique maximale.

Définition 3.2.9 *Soit Σ un alphabet et R une relation binaire sur $\Sigma \times \Sigma$. Soit C un sous-ensemble de Σ. On appelle C une clique de R si, pour tout $a, b \in C$, on a $a\ R\ b$. En d'autres termes, C est une clique si tous ses éléments sont 2 à 2 en relation.*

Si la relation R est représentée par son graphe G, alors les cliques de R sont tous les sous-graphes complets de G.

Définition 3.2.10 *Soit Σ un alphabet, R une relation sur $\Sigma \times \Sigma$ et C une clique de R. C est dit être une clique maximale si, pour tout $a \in \Sigma \backslash C$, $C \cup \{a\}$ n'est plus une clique.*

En termes du graphe G représentant R, les cliques maximales de R sont tous les sous-graphes complets maximaux de G.

Un exemple d'une telle relation et de ses cliques maximales est donnée dans la figure 3.9.

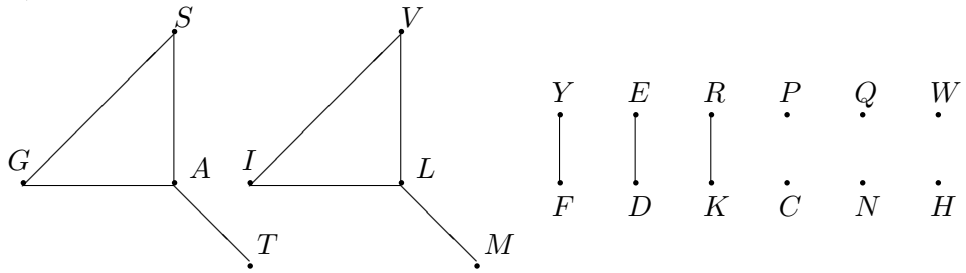
Cette représentation de la mesure de ressemblance entre deux monomères permet ainsi une extension au cas de plus de deux éléments dont nous allons voir plus loin qu'elle est à la fois très satisfaisante du point de vue conceptuel, et conduit à des algorithmes de comparaison de chaînes de caractères qui sont efficaces.

Avant de clore cette section, observons qu'à partir de la notion de clique maximale, il est possible de définir un concept qui sera utilisé par la suite et qui mesure l'indice de non-transitivité d'une relation R de similarité par rapport à l'identité :

Définition 3.2.11 *Soit R une relation sur Σ et $a \in \Sigma$, on note g_a le nombre de cliques maximales auxquelles a appartient.*

Figure 3.8: Une illustration de la notion d'équivalence.

Soit $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ l'alphabet des acides aminés et R la relation de similarité entre ces acides aminés obtenue en seuillant (à la valeur 17) la matrice PAM de la figure 3.7 et schématisée par le graphe suivant :



Les cliques maximales de R sont les ensembles :
$\{A, S, G\}, \{A, T\}, \{I, L, V\}, \{L, M\}, \{F, Y\}, \{D, E\}, \{K, R\}, \{C\}, \{P\}, \{N\}, \{Q\}, \{W\}, \{H\}$.

Figure 3.9: Exemple d'une relation de similarité entre les acides aminés et de ses cliques maximales.

Définition 3.2.12 *Soit R une relation de Σ, on appelle g le plus grand nombre de cliques maximales de R à laquelle un symbole de Σ appartient, à savoir :*

$$g = Max\ \{g_a \mid a \in \Sigma\}.$$

On appelle $\overline{g}$ la valeur moyenne de g_a pour $a \in \Sigma$, c'est-à-dire :

$$\overline{g} = \frac{\sum_a g_a}{\mid \Sigma \mid}.$$

Dans l'exemple de R donné dans la figure 3.9, g est égal à 2 et $\overline{g}$ à 1.1. L'indice de non-transitivité d'une relation d'équivalence est la même que celle de l'identité, à savoir : $g = 1$.

3.2.3.2.2 Relation sur l'alphabet des objets 3D — Graphe d'intervalles

Dans le cas où les monomères sont considérés en tant qu'objets dans l'espace, plus spécialement en tant que résidus le long d'une chaîne qui se replie sur elle-même pour former une structure tridimensionnelle, nous avons vu (section 3.2.2.3) qu'il est possible de les associer de manière univoque à des paires d'angles, qui elles-mêmes sont codées sur des symboles qui sont les numéros des carrés d'une grille d'échantillonnage. Ces symboles (les numéros) représentent un nouvel alphabet sur lequel la structure de la protéine est définie et une relation R est alors établie entre ces symboles (noeuds de la carte) de la façon suivante :

$$\forall\ (\alpha, \beta) \in \Sigma^2,\ \alpha\ R\ \beta \text{ si et seulement si } \exists \text{ un carré de côté } 2K\epsilon \text{ qui contient } \alpha \text{ et } \beta$$

où α et β sont les noeuds de la grille et K est un paramètre (appelé marge) qui peut être ajusté pour augmenter ou diminuer la tolérance. Comme pour une relation sur l'alphabet des monomères, nous pouvons établir l'indice de non-transitivité de la relation R sur l'alphabet des

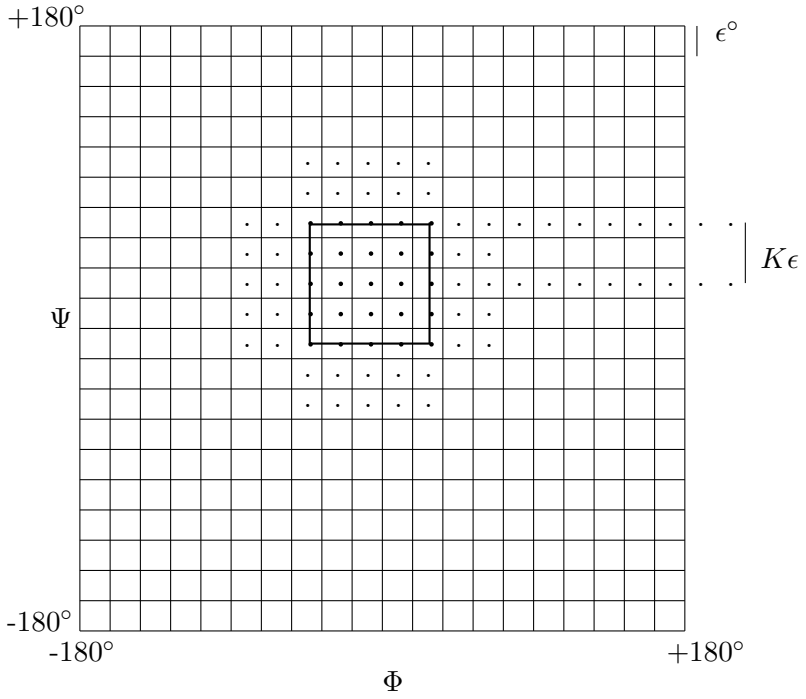


Figure 3.10: Carte de Ramachandran échantillonée : cliques maximales de la relation $R =$ grands carrés de taille $K\epsilon$.

objets 3D. Diminuer les valeurs de ϵ et K conduit à des alphabets et des relations de tailles et indices de non-transitivité croissants respectivement (pour une maille de 5° par exemple l'alphabet possède 5182 symboles; et pour $K = 1$, g est égal à 9). En termes d'angles, la définition précédente signifie simplement que deux paires d'angles sont équivalentes si elles sont égales à plus ou moins ($\Delta\Phi = K\epsilon, \Delta\Psi = K\epsilon$). Bien que la relation R soit définie sur les ensembles de noeuds de la grille et non sur les paires d'angles elles-mêmes, elle possède la même caractéristique intrinsèquement non transitive de la relation de similarité entre angles et correspond à la relation associée à un graphe d'intervalles. Les cliques maximales de R sur Σ sont alors par définition tous les grands carrés de taille $K\epsilon$ centrés sur chaque noeud (voir la figure 3.10). Nous montrons plus loin comment cette relation R peut être étendue aux mots structuraux.

3.2.3.3 Couverture

3.2.3.3.1 Couverture sur l'alphabet des monomères

Initialement, le concept de couverture que nous utilisons ici a trouvé sa motivation dans la constatation qu'une seule relation n'est souvent pas suffisante pour capter toute la richesse des liens existants entre les acides aminés. La complexité de ces liens se trouve alors mieux illustrée

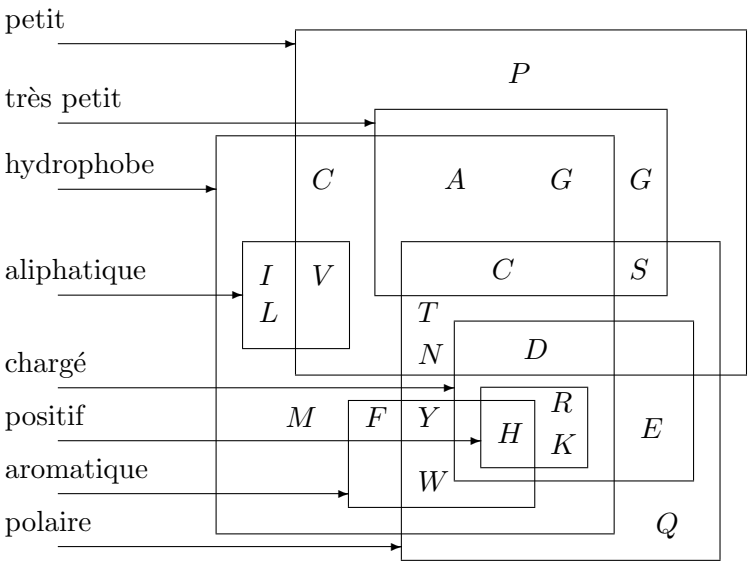


Figure 3.11: Diagramme Venn proposé par Taylor pour les acides aminés.

par un diagramme de Venn comme celui proposé par Taylor [Taylor, 1986a] et montré dans la figure 3.11. Ce diagramme a été établi à la fois à partir de données mutationnelles comme celles présentes dans une des matrices que nous avons décrites (dans son cas la matrice utilisée est celle de Dayhoff) et directement des propriétés physico-chimiques des acides aminés.

Une couverture est l'objet le mieux adapté pour modéliser ces liens [Sagot *et al.*, 1995a] [Sagot *et al.*, 1995d] [Sagot *et al.*, 1996].

Définition 3.2.13 *Soit Σ un alphabet, une couverture de Σ est un ensemble de sous-ensembles S_1, S_2, ... , S_p de Σ tels que :*

$$\bigcup_{i=1}^{p} S_i = \Sigma.$$

Sans recevoir un nom spécifique, cet objet a été utilisé par plusieurs personnes avant nous [Galas *et al.*, 1985] [Karlin and Ghandour, 1985b] [Karlin and Ghandour, 1985a] [Neuwald and Green, 1994] [Smith and Smith, 1990] [Waterman, 1984b] [Waterman *et al.*, 1984] [Waterman, 1989]. Dans tous les cas à l'exception de Neuwald, la couverture en question est en fait une partition.

Un exemple d'une couverture qui n'est pas une partition et qui s'inspire du diagramme de Taylor est donné dans la figure 3.12.

Observons qu'une couverture C peut être représentée par un hypergraphe sur l'alphabet Σ. Par ailleurs, nous parlons de l'indice de non-transitivité g d'une couverture de la même façon que nous avons parlé de l'indice de non-transitivité d'une relation R. Formellement :

Définition 3.2.14 *Soit C une couverture de Σ et $a \in \Sigma$, on note g_a le nombre d'ensembles de C auxquels a appartient.*

La couverture donnée ci-dessous est fortement inspirée du regroupement proposé par Taylor [Taylor, 1986a] et indiqué dans la figure 3.11.
$S_1 = \{A, C, G, S, T\}$ — très petit
$S_2 = \{A, C, D, G, N, S, T, V\}$ — petit
$S_3 = \{F, H, W, Y\}$ — aromatique
$S_4 = \{V, L, I, M\}$
$S_5 = \{K, R, H\}$ — basique
$S_6 = \{D, E\}$ — acide
$S_7 = \{Q, N\}$ — amide
$S_8 = \{P\}$ — proline
Observez que cette couverture ne représente ni une partition de Σ ni les cliques maximales d'une relation.

Figure 3.12: Exemple d'une couverture de l'alphabet des acides aminés.

Définition 3.2.15 *Soit C une couverture de Σ, on appelle g le nombre maximum d'ensembles de C auquel un symbole de Σ appartient, à savoir :*

$$g = Max\ \{g_a \mid a \in \Sigma\}.$$

On appelle par ailleurs $\overline{g}$ la valeur moyenne de g_a pour $a \in \Sigma$, c'est-à-dire :

$$\overline{g} = \frac{\sum_a g_a}{\mid \Sigma \mid}.$$

Dans l'exemple de la figure 3.12, g est égal à 2 et $\overline{g}$ à 1.45. g mesure ainsi la non-transitivité de la couverture par rapport à ce que nous pouvons appeler la couverture identité définie par :

Notation 3.2.2 *On note CI la couverture identité égale à $\{\{a\} \mid a \in \Sigma\}$.*

La couverture identité est celle habituellement employée pour les nucléotides mais ce n'est pas la seule ainsi que nous l'indiquons à la fin de cette section.

Lorsque C représente en fait une partition, sa non-transitivité est la même que celle de CI, c'est-à-dire 1.

Observons en outre qu'à nouveau ici la mesure de ressemblance est discrétisée car deux monomères sont considérés comme étant soit similaires s'il existe un ensemble de la couverture qui les contient tous deux, soit non similaires si cela n'est pas le cas. Comme pour les cliques, une couverture permet également une formalisation très satisfaisante de la ressemblance entre plus de deux objets à la fois. Le rapport existant entre clique et clique maximale d'un côté et couverture d'un autre va plus loin puisque l'ensemble des cliques maximales d'une relation forme une couverture.

La réciproque cependant n'est pas vraie. Le diagramme de Taylor en particulier ne peut être obtenu par simple seuillage d'une matrice comme c'est le cas pour la relation de la section précédente puisqu'en fait il exprime plusieurs relations à la fois. Il permet ainsi par exemple de caractériser un ensemble d'acides aminés à deux niveaux différents de généralité, les plaçant d'abord dans une même unité qui ne contient qu'eux, puis les groupant avec un ou plusieurs autres acides aminés qui possèdent les mêmes attributs que les éléments de l'ensemble plus petit, plus quelques autres qui leur sont propres.

Figure 3.13: Points de vue différents.

L'information captée par une couverture peut être encore plus subtile. Illustrons cela par un exemple. Imaginons que l'on veuille comparer non plus les monomères d'une macromolécule biologique, mais certains animaux, par exemple des lapins, des baleines et des sardines. Pour un zoologiste, un lapin est plus proche d'une baleine que les deux d'une sardine car tous deux sont des mammifères et l'attribut 'être un mammifère' est important en zoologie. Pour le patron d'une entreprise alimentaire par contre, il est sans doute plus cohérent de classer la baleine avec la sardine car bien que les trois animaux soient (en principe) comestibles, la baleine et la sardine proviennent du même milieu, la mer, et c'est sans doute un des facteurs les plus importants pour lui en termes de l'infrastructure dont il a besoin pour installer son industrie (traiter des lapins exige d'autres moyens que ceux nécessaires pour traiter les produits de la pêche). Finalement, un enfant souhaitant adopter un animal n'hésitera pas à classer le lapin avec la sardine car il sait que tous deux peuvent entrer dans son appartement, une baleine non. Ces points de vue divers donnent ainsi lieu à trois classifications (voir la figure 3.13) qu'une couverture reproduit fidèlement mais que le concept de clique maximale gommerait.

Cette image sert donc à illustrer encore une fois le fait que les critères permettant de mesurer la ressemblance de deux objets dépendent de ce que nous recherchons. Ces critères sont très

probablement différents suivant que nous désirons identifier par exemple l'emplacement des sites actifs d'une protéine, ou des zones d'interaction sur l'ADN et l'ARN, ou encore établir les facteurs à partir desquels une chaîne polymérique se replie dans l'espace pour produire une structure. Ils sont différents aussi suivant le temps depuis lequel les molécules que nous voulons comparer ont divergé à partir d'un ancêtre commun. Le problème est que ces critères ne sont pas toujours connus, et ceux qui le sont n'ont probablement pas un caractère définitif.

Parfois alors il faudrait pouvoir aller plus loin et jouer sans *a priori* avec plusieurs points de vue différents à la fois, aussi bien dans le cas de l'ADN et de l'ARN que dans celui des protéines. Un premier outil pour réaliser cela va constituer ce que nous désignons sous le nom de couverture combinatoire pondérée. Dans sa définition la plus complète, une couverture combinatoire pondérée CCP est simplement l'ensemble des parties de l'alphabet avec une pondération affectée à chacun de ces sous-ensembles. La signification précise de ces contraintes deviendra claire lorsque nous passerons de la comparaison des perles à celles des colliers. Pour l'instant, il suffit de dire qu'elles sont nécessaires afin d'éviter que tout ou presque tout finisse par se ressembler. Nous laissons donc pour plus tard leur définition et une discussion des limites conceptuelles de cette approche.

Observons cependant dès maintenant que, même lorsque nous aurons établi des contraintes sur ces couvertures combinatoires, la souplesse qu'elles autorisent vont exiger aussi des raffinements afin de nous permettre de les utiliser d'une manière qui soit algorithmiquement efficace. Ces raffinements sont formellement intéressants en eux-mêmes et peut-être extensibles à des problèmes d'une autre nature. Ils sont présentés au chapitre 5.

3.2.3.3.2 Couverture sur l'alphabet des objets 3D

Notons simplement ici que le codage de la structure d'une macromolécule en tant que chaîne sur des symboles qui représentent les numéros des carrés d'une grille discrétisants des paires d'angles peut *a priori* conduire à une mesure de la ressemblance utilisant le concept d'une couverture. Les éléments de cette couverture seraient comme auparavant des ensembles de carrés (numéros) proches les uns des autres mais comme il n'y a plus ici de notion de maximalité, ces ensembles pourraient être représentés non seulement par des grands carrés de taille $K\epsilon$ mais également par n'importe quel autre découpage de la grille.

3.2.4 La ressemblance des colliers de perles revue

3.2.4.1 Valeur d'un alignement

3.2.4.1.1 Alignement sans trous

Nous allons vouloir à présent passer d'une mesure de la ressemblance entre perles à une mesure de la ressemblance entre colliers alignés.

Commençons par supposer que nous avons réalisé un alignement de deux molécules et, pour simplifier, considérons pour l'instant le cas où cet alignement ne comporte pas de trous. Quelle valeur lui affecter?

Il y a deux façons de procéder qui représentent en fait deux philosophies différentes de la mesure de ressemblance mais qui peuvent être formalisées de manière unique puisque dans les deux cas, la valeur attribuée à l'alignement, appelée un score, est égale à la somme des scores affectés aux paires alignées. Ces derniers cependant peuvent représenter soit une valeur numérique, entière ou réelle, trouvée dans une matrice de substitution, soit une valeur booléenne dans le cas d'une relation ou d'une couverture.

Notation 3.2.3 *Soient $a_1a_2...a_n$ et $b_1b_2...b_n$ deux chaînes sur un alphabet Σ. La suite définie sur $\Sigma \times \Sigma$ égale à $(a_1, b_1)(a_2, b_2)...(a_n, b_n)$ et notée $\begin{pmatrix} a_1 \\ b_1 \end{pmatrix}\begin{pmatrix} a_2 \\ b_2 \end{pmatrix}...\begin{pmatrix} a_n \\ b_n \end{pmatrix}$ représente l'alignement sans trous des deux chaînes (dans ce cas un seul alignement est possible).*

Formellement nous avons :

Définition 3.2.16 *Soit $\begin{pmatrix} a \\ b \end{pmatrix}$ un alignement de deux symboles de Σ. Alors la valeur de score $\begin{pmatrix} a \\ b \end{pmatrix}$ est donnée par :*

- $\mathcal{M}(a, b)$ *pour une matrice de substitution $\mathcal{M}$;*
- $\begin{cases} 1 \text{ si } aR\,b \\ 0 \text{ sinon} \end{cases}$ *pour une relation de similarité R;*
- $\begin{cases} 1 \text{ si } \exists S \in C \text{ tel que } a, b \in S \\ 0 \text{ sinon} \end{cases}$ *pour une couverture C de l'alphabet.*

Définition 3.2.17 *Soit $\mathcal{A} = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}\begin{pmatrix} a_2 \\ b_2 \end{pmatrix}...\begin{pmatrix} a_n \\ b_n \end{pmatrix}$ un alignement des deux chaînes (définies sur un alphabet Σ) $a_1a_2...a_n$ et $b_1b_2...b_n$. Alors :*

$$score\mathcal{A} = \sum_{i=1}^{n} score\begin{pmatrix} a_i \\ b_i \end{pmatrix}.$$

Exemple 3.2.1 *Soient les deux chaînes $s1$ et $s2$ définies sur l'alphabet $\Sigma = \{a, b, c, d\}$ et données par :*
$s1 = abcbdcb$
$s2 = acbbdca$.
Supposons que $\mathcal{M}_S = \begin{pmatrix} 3 & -1 & 0 & -2 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -2 & 0 & 0 & 4 \end{pmatrix}$, $R = I$ (voir la notation 3.2.1), $C = CI$ (voir la notation 3.2.2).
Alors le score de l'alignement $\mathcal{A} = \begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} b \\ c \end{pmatrix}\begin{pmatrix} c \\ b \end{pmatrix}\begin{pmatrix} b \\ b \end{pmatrix}\begin{pmatrix} d \\ d \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} b \\ a \end{pmatrix}$ sans trous de $s1$ et $s2$ est égal à 7 si nous utilisons une matrice, et à 4 si nous travaillons avec une relation ou une couverture.

Si la matrice utilisée est une matrice de dissimilarité (resp. de distance), alors le score de $\mathcal{A}$ représente en fait une dissimilarité (resp. une distance) entre les deux chaînes. Dans le cas d'une mesure entre monomères ayant pour base une relation ou une couverture et où les valeurs données dans la définition 3.2.16 sont inversées (i.e. 0 si a R b ou si $\exists$ $S \in C$ tel que a, $b \in S$, 1 sinon), le score de $\mathcal{A}$ indique une distance. Si de plus $R = I$ ou $C = CI$, cette distance représente la distance de Hamming entre les deux chaînes.

Exemple 3.2.2 *La distance de Hamming entre les chaînes $s1$ et $s2$ de l'exemple 3.2.1 est égale à 3.*

La définition précédente peut être réécrite de manière récursive. Cette façon de faire va nous être particulièrement utile par la suite. En effet, nous allons voir que la récursion est à la base de presque toutes les méthodes d'exploration de la ressemblance que nous présentons ici.

Définition 3.2.18 *Soit* $\mathcal{A} = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}\begin{pmatrix} a_2 \\ b_2 \end{pmatrix}...\begin{pmatrix} a_n \\ b_n \end{pmatrix}$ *un alignement des deux chaînes (définies sur un alphabet* Σ*)* $a_1a_2...a_n$ *et* $b_1b_2...b_n$ *avec* $n \geq 1$*. Alors :*

$$score\mathcal{A} = score\mathcal{A}' + score\begin{pmatrix} a_n \\ b_n \end{pmatrix}$$

où $\mathcal{A}' = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix}\begin{pmatrix} a_2 \\ b_2 \end{pmatrix}...\begin{pmatrix} a_{n-1} \\ b_{n-1} \end{pmatrix}$.

Pourquoi avons-nous dit qu'il y avait en fait ici deux philosophies différentes de la mesure de ressemblance entre deux objets suivant que nous travaillons avec une matrice ou avec une relation ou couverture? Prenons l'exemple des examens que doivent passer les étudiants. La réussite à un cours peut être déterminée d'au moins deux manières : soit une moyenne (éventuellement pondérée) est établie des notes obtenues sur l'ensemble des disciplines que l'étudiant doit suivre et ce dernier est reçu si cette moyenne est au-dessus de 10, soit pour être reçu celui-ci doit obtenir une note au-dessus de 10 dans au moins un certain nombre x de disciplines. Tout le monde sait pourquoi la seconde façon de faire peut être plus sévère : il n'y a pas de rattrapage possible des notes. Si vous obtenez 19 dans une discipline, vous l'avez bien sûr réussie, mais vous ne l'avez pas plus réussie que si vous aviez obtenu 10. Et si vous obtenez 9, vous l'avez tout aussi bien ratée que si vous aviez obtenu 0.

Le résultat est le même dans le cas qui nous concerne et la première manière de procéder, utilisant une matrice, est ainsi plus flexible que la seconde. Il est important d'observer que deux facteurs sont en jeu ici : l'effet de compensation et l'effet de pondération. Le second permet d'attribuer des poids différents à une même propriété suivant les objets qui la vérifient. Par exemple, l'appariement de deux monomères identiques dans un alignement peut recevoir un score variable suivant la nature du monomère en question. Un acide aminé plus rare a ainsi un score d'alignement avec lui-même plus élevé indiquant son plus faible taux de mutabilité. L'effet de compensation quant à lui est une façon détournée de prendre en considération le contexte d'un appariement, du moins en partie. Il permet ainsi, selon la définition de ressemblance établie, de moduler le score d'un mauvais appariement par les scores des appariements avoisinants dans un alignement. Cette modulation n'est plus possible lorsque l'on travaille avec une relation R ou une couverture C. Une autre façon de voir cela est de considérer que travailler avec ces deux concepts revient à travailler avec un système de score qui attribue la valeur 1 à un appariement de deux monomères en relation par R (ou qui appartiennent à un même sous-ensemble de C) et -∞ sinon. Les deux effets, de pondération et de compensation, sont illustrés dans la figure 3.14 dans le cas d'une chaîne de caractère. Cette plus grande flexibilité permise par l'utilisation d'un système de score peut cependant être aussi plus délicate à manier et à interpréter. Les deux approches sont en fait valables, et nous laissons pour le chapitre consacré aux illustrations une discussion de leurs mérites respectifs ainsi que des problèmes que chacune présente.

Observons toutefois avant de clore cette section qu'il n'est en fait pas obligatoire d'attribuer la valeur 1 à une paire de symboles en relation ou appartenant à un même ensemble d'une couverture. Nous pourrions en effet utiliser une valeur w_i qui dépendrait de la clique maximale C_i de R ou de l'ensemble S_i de la couverture C. Notons que, dans le cas d'une relation, ces scores w_i seraient définis après l'établissement de la relation, qui elle-même aurait pu être obtenue à

Soit $\Sigma = \{a, b, c\}$ et soient :

$$\mathcal{M}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ et } \mathcal{M}_2 = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}.$$

Si nous établissons que deux chaînes sont similaires si :
- le nombre d'identités est ≥ 5;
ou
- le score d'un alignement sans trou des deux est ≥ 5;

alors par le premier critère les chaînes *aacbcba* and *aaabaaa* ne sont pas similaires tandis que par le second elles le sont que la matrice utilisée soit $\mathcal{M}_1$ ou $\mathcal{M}_2$. La première illustre l'effet de pondération appliqué sur une même propriété (ici l'identité de deux monomères) et la seconde l'effet de compensation, les 3 mauvais appariements étant 'rattrapés' localement par l'alignement des paires identiques.

Figure 3.14: Exemple de la plus grande flexibilité que permet l'utilisation de scores sur un simple comptage du nombre d'erreurs ou d'identités.

partir du seuillage d'une matrice de scores. Les scores w_i et les scores de la matrice servant éventuellement à définir R ou C n'ont ainsi pas besoin d'être identiques. L'établissement de poids w_i dépendants de C_i ou de S_i peuvent alors introduire un effet de pondération et de compensation comme dans le cas des matrices, mais la nature de ces effets est différente. Celui de pondération interviendrait au niveau de groupes d'objets (des cliques maximales ou non) plutôt qu'au niveau des objets individuels. Celui de compensation quant à lui ne serait observé que si un score négatif mais non infini était attribué aux paires de symboles non en relation par R ou appartenant à des éléments de C différents. Ces scores négatifs pourraient eux-mêmes dépendre de quelles cliques de R ou éléments de C sont impliqués par la paire appariée. Il n'est pas du tout clair cependant comment les poids w_i ainsi que les poids de mésappariement pourraient être déterminés. Aucune théorie n'existe dans ce sens, même sujette à précaution comme celle associée à la construction des matrices. Relation et couverture sont donc utilisés de manière discrétisée et le score que nous employons en pratique est celui indiqué dans la définition 3.2.16.

Nous allons faire l'hypothèse maintenant que nous disposons d'un alignement des deux molécules qui présente des trous. Voyons comment lui attribuer une valeur.

3.2.4.1.2 Alignement avec trous

Un trou dans un alignement de deux molécules ayant même origine signifie qu'une mutation correspondant soit à une délétion, soit à une insertion a eu lieu. De tels événements sont considérés plus rares que les substitutions non pas parce que les délétions et insertions surviennent moins souvent, mais parce qu'elles sont observées moins souvent. Nous avons vu dans le chapitre précédent que ces mutations en effet conduisent plus fréquemment à une perte d'activité de la macromolécule. Toute la difficulté d'attribuer une valeur aux trous présents

dans un alignement vient alors de cette constatation. En effet, les méthodes qui essayent de déterminer quelle pondération il faudrait affecter à un trou sont soit *ad hoc*, soit emploient les mêmes approches que celles utilisées pour calculer certaines matrices de substitution : c'est-à-dire, par une observation des délétions ou insertions survenues entre macromolécules homologues ([Benner *et al.*, 1993] [Pascarella and Argos, 1992] pour les protéines). C'est comme si pour estimer la qualité de l'alimentation des jeunes français des années trente, on décidait de mesurer l'état de santé des personnes âgées entre soixante-quinze et quatre-vingt-dix ans aujourd'hui. Même si nous laissons de côté la question de comment effectivement quantifier une telle qualité qui fasse la part des multiples autres facteurs pouvant avoir une influence sur la santé des individus, une telle mesure ne peut être qu'un pauvre reflet de la qualité réelle de cette alimentation puisqu'elle néglige toutes les personnes décédées entre temps pour diverses raisons (dont quelqu'unes peuvent être justement liées à leur alimentation).

Outre la difficulté que nous venons d'illustrer, et qui est insurmontable, l'attribution d'une valeur à un trou dans un alignement pose deux autres types de problèmes. Le premier est semblable à celui rencontré pour l'affectation d'une mesure à la ressemblance de deux monomères. Il n'est pas possible d'attribuer une seule valeur à un trou quel que soit son contexte. Dans le cas d'une protéine par exemple, une délétion ou insertion observée à l'intérieur d'une structure secondaire ou dans une région correspondant à un site ne peut pas avoir le même poids qu'une délétion ou insertion observée ailleurs, soit à une des extrémités de la molécule, soit dans les parties moins structurées ou qui ne sont pas impliquées dans son activité. Le poids doit varier également suivant les caractéristiques physico-chimiques de l'acide aminé qui a été délété ou inséré. L'insertion d'un 'gros' acide aminé par exemple peut en effet rompre localement ou globalement la structure d'une chaîne, alors que celle d'un 'petit' résidu comme la glycine ne modifie éventuellement que peu cette structure et peut être tolérée. Toute la difficulté réside à nouveau ici dans le fait que ce contexte est rarement connu.

L'attribution d'une valeur à un trou pose un second problème qui concerne sa longueur cette fois. Si une substitution est un événement ponctuel, une délétion ou une insertion implique souvent plusieurs résidus en même temps. Il n'est pas nécessairement question ici de corrélations entre mutations (bien que cela puisse également être le cas) mais d'un unique événement mutationnel affectant une région plus ou moins étendue. Ceci correspond donc à une suite de trous dans un alignement. Il n'est pas admissible d'attribuer une valeur constante à chacun d'eux puisqu'ils se réfèrent à un même phénomène survenu une seule fois. Ici aussi, plusieurs manières de procéder ont été proposées, la plus usuelle consiste à pénaliser plus fortement le premier trou. Le score d'une suite de trous est ainsi une fonction affine de sa longueur, c'est-à-dire est égale à $ak + b$ où k est la longueur de la suite, a et b deux constantes [Altschul and Erickson, 1986] [Gotoh, 1982]. Un cas particulier de cette fonction est obtenue pour $a = 0$, le score d'un trou est alors lui-même une constante [Sellers, 1974] [Ukkonen, 1985a]. D'autres fonctions ont été imaginées, par exemple, des fonctions affines par morceaux [Gotoh, 1990] [Miller and Myers, 1988] où la valeur de a change par intervalles, ou encore des fonctions concaves [Fitch and Smith, 1983] [Galil and Giancarlo, 1989] [Miller and Myers, 1988] [Miller and Myers, 1988] [Waterman, 1984a]. Gonnet et Benner ont toutefois suggéré que les fonctions affines ne seraient pas un modèle valable pour les alignements protéiques. D'après les résultats de leur analyse de la distribution des trous dans les alignements de protéines homologues [Benner *et al.*, 1993], le poids à accorder à l'ouverture d'un trou varie avec la distance évolutive des protéines tandis que celui à affecter à l'allongement d'un trou, bien qu'indépendant de cette distance, augmente non pas linéairement avec la longueur mais en fonction de cette longueur élevée à la puissance 3/2. Ces résultats ne correspondent cependant pas tout à fait à ceux obtenus par Pascarella [Pascarella and Argos, 1992].

Nous voyons ainsi que les controverses autour de la valeur à attribuer à l'introduction d'un trou ou à son allongement sont encore plus grandes que celles concernant les substitutions. Ces controverses sont importantes, il a été montré que la qualité d'un alignement dépend en effet plus fortement du choix des pénalités affectées aux trous que de celui des scores de substitutions [Gonnet *et al.*, 1992]. Différents travaux ont porté aussi sur les rapports existant entre ce choix et celui du meilleur alignement qu'il est possible d'obtenir entre deux chaînes [Gusfield *et al.*, 1992] [Vingron and Waterman, 1994] [Waterman *et al.*, 1992].

Cependant notre propos n'est pas de les discuter ici. La raison principale en est que nous nous intéressons plus dans ce travail aux mots qu'aux chaînes entières, et que ces mots représentent comme nous l'avons vu les régions les mieux conservées des macromolécules. Il n'est pas abusif de considérer que, dans ce cas, le choix d'un score à attribuer aux trous n'est pas aussi crucial. En effet, la présence de trous y est observée moins fréquemment que sur le reste de la molécule, et ces trous correspondent le plus souvent à des événements mutationnels ponctuels, les délétions ou insertions de blocs entiers se produisant plutôt entre ces régions. Compter simplement chacun d'eux ou leur attribuer un poids constant apparaît en grande partie justifié dans ce cadre [Altschul, 1989]. Nous allons donc admettre ce choix pour l'instant et voir quel est alors le score d'un alignement avec trous.

Notation 3.2.4 *Soient* $a_1a_2...a_n$ *et* $b_1b_2...b_m$ *deux chaînes sur un alphabet* Σ *avec* $n \leq m$ *et soit* $\overline{\Sigma} = \Sigma \cup \{\lambda\}$. *La suite définie sur* $\overline{\Sigma} \times \overline{\Sigma}$ *égale à* $\begin{pmatrix} \overline{a}_1 \\ \overline{b}_1 \end{pmatrix} \begin{pmatrix} \overline{a}_2 \\ \overline{b}_2 \end{pmatrix} ... \begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix}$ *avec* $m \leq p \leq n+m$ *et telle que :*

- $\overline{a}_i = a_j$ *ou* $\overline{a}_i = \lambda$ *pour* $1 \leq i \leq p$ *et* $1 \leq j \leq n$
- $\overline{b}_i = b_j$ *ou* $\overline{b}_i = \lambda$ *pour* $1 \leq i \leq p$ *et* $1 \leq j \leq m$
- $| \overline{a}_1...\overline{a}_p | = n$ *et* $\overline{a}_1...\overline{a}_p = a_1...a_n$
- $| \overline{b}_1...\overline{b}_p | = m$ *et* $\overline{b}_1...\overline{b}_p = b_1...b_m$
- *pour tout* $i \in \{1,...,p\}$, *il n'est jamais vrai que* $\overline{a}_i = \lambda = \overline{b}_i$ *(pas de trou mis face à un trou)*

représente un alignement de $a_1a_2...a_n$ *avec* $b_1b_2...b_m$.

Exemple 3.2.3 *Soient les deux chaînes* $s1$ *et* $s2$ *définies sur l'alphabet* $\{a, b, c, d\}$ *et données par :*

$s1 = abcbadcb$

$s2 = acbbdca$.

Alors $\begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} b \\ \lambda \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} b \\ b \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}\begin{pmatrix} d \\ d \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} b \\ a \end{pmatrix}$ *est un alignement de* $s1$ *et* $s2$.

Si l'on considère la dernière paire d'un alignement $\begin{pmatrix} \overline{a}_1 \\ \overline{b}_1 \end{pmatrix} \begin{pmatrix} \overline{a}_2 \\ \overline{b}_2 \end{pmatrix} ... \begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix}$ de deux chaînes $a_1a_2...a_n$ et $b_1b_2...b_m$, trois cas de figures sont alors possibles :

- $\begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix} = \begin{pmatrix} a_n \\ b_m \end{pmatrix}$
- $\begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix} = \begin{pmatrix} a_n \\ \lambda \end{pmatrix}$

- $\begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix} = \begin{pmatrix} \lambda \\ b_m \end{pmatrix}$.

Nous utilisons ce fait pour donner une définition récursive du score d'un alignement $\mathcal{A}$.

Définition 3.2.19 *Soit* $\begin{pmatrix} \overline{a} \\ \overline{b} \end{pmatrix}$ *un alignement de deux symboles de* $\Sigma \cup \{\lambda\}$ *dont l'un des deux au moins est différent de* λ. *Alors, score* $\begin{pmatrix} \overline{a} \\ \overline{b} \end{pmatrix}$ *est égal à :*

- $\begin{cases} \mathcal{M}(a,b) \text{ si } \overline{a} = a \neq \lambda \text{ et } \overline{b} = b \neq \lambda \\ w_\lambda \text{ sinon} \end{cases}$ *pour une matrice* $\mathcal{M}$ *et une constante* w_λ

  (w_λ *représente la pénalité à attribuer à un trou);*

- $\begin{cases} 1 \text{ si } \overline{a} = a \neq \lambda \text{ et } \overline{b} = b \neq \lambda \text{ et } aR\,b \\ 0 \text{ sinon} \end{cases}$ *pour une relation de similarité* R*;*

- $\begin{cases} 1 \text{ si } \overline{a} = a \neq \lambda \text{ et } \overline{b} = b \neq \lambda \text{ et } \exists S \in C \text{ tel que } a, b \in S \\ 0 \text{ sinon} \end{cases}$ *pour une couverture* C.

Définition 3.2.20 *Soit* $\mathcal{A} = \begin{pmatrix} \overline{a}_1 \\ \overline{b}_1 \end{pmatrix}\begin{pmatrix} \overline{a}_2 \\ \overline{b}_2 \end{pmatrix}...\begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix}$ *un alignement des chaînes* $a_1a_2...a_n$ *et* $b_1b_2...b_m$. *Alors :*

$$score\mathcal{A} = score\mathcal{A}' + score\begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix}$$

où $\mathcal{A}' = \begin{pmatrix} \overline{a}_1 \\ \overline{b}_1 \end{pmatrix}\begin{pmatrix} \overline{a}_2 \\ \overline{b}_2 \end{pmatrix}...\begin{pmatrix} \overline{a}_{p-1} \\ \overline{b}_{p-1} \end{pmatrix}$.

Exemple 3.2.4 *Soient les deux chaînes* $s1$ *et* $s2$ *définies sur l'alphabet* $\Sigma = \{a, b, c, d\}$ *et données par :*

$s1 = abcbadcb$

$s2 = acbbdca$.

Supposons que $\mathcal{M}_S = \begin{pmatrix} 3 & -1 & 0 & -2 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -2 & 0 & 0 & 4 \end{pmatrix}$, $R = I$ *(voir la notation 3.2.1),* $C = CI$ *(voir la notation 3.2.2) et* w_λ = -1, *Alors le score de* $\mathcal{A} = \begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} b \\ \lambda \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} b \\ b \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}\begin{pmatrix} d \\ d \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} b \\ a \end{pmatrix}$ *est égal à* 8 *si nous utilisons une matrice et à* 5 *si nous travaillons avec une relation ou une couverture.*

Dans le cas d'une relation ou d'une couverture, il est habituel de travailler avec une mesure de dissimilarité plutôt qu'avec une mesure de similarité, et ainsi la valeur d'un alignement est donné en inversant les rôles de 0 et 1 dans la définition 3.2.19. Lorsque $R = I$ ou $C = CI$ et que score $\mathcal{A} \leq$ score $\mathcal{B}$ pour tout alignement $\mathcal{B}$ de $a_1a_2...a_n$ avec $b_1b_2...b_m$, alors le score de $\mathcal{A}$ mesure en fait la distance d'édition classique que nous avons mentionnée au début de ce

chapitre. Elle correspond bien au plus petit nombre d'opérations de substitutions, délétions et insertions nécessaires pour passer d'une chaîne à l'autre. Nous appelons aussi cette distance la distance de Levenshtein [Levenshtein, 1966] séparant deux chaînes de caractères et nous utilisons indifféremment les deux termes dans le reste de ce travail.

Exemple 3.2.5 *La distance de Levenshtein des chaînes $s1$ et $s2$ de l'exemple 3.2.4 est égale à 3.*

3.2.4.2 Mesure d'une ressemblance : la valeur du meilleur alignement

Nous venons de montrer comment attribuer un score à un alignement. Une mesure de la ressemblance entre objets décomposables en unités élémentaires sur lequel un ordre total a pu être établi est alors classiquement donnée par le plus grand (ou le plus petit) score obtenu parmi tous les alignements possibles de ces objets (suivant que l'on considère une similarité ou une dissimilarité entre eux).

Définition 3.2.21 *Soient $s1$ et $s2$ deux chaînes. Soit $VR(s1, s2)$ la valeur de similarité (resp. dissimilarité) entre $s1$ et $s2$, alors :*

$$VR(s1, s2) = max\ (resp.min)\ \{score\mathcal{A} \mid \mathcal{A}\ un\ alignement\ de\ s1\ et\ s2\}.$$

Exemple 3.2.6 *En reprennant les données de l'exemple 3.2.4, alors $VR(s1, s2)$ est égal à 8 ou 5 suivant que nous travaillons avec une matrice ou avec une relation/couverture puisque l'alignement $\mathcal{A} = \begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} b \\ \lambda \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} b \\ b \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}\begin{pmatrix} d \\ d \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} b \\ a \end{pmatrix}$ sans trou représente un meilleur alignement de $s1$ et $s2$.*

Cette mesure correspond ainsi au score du meilleur alignement de $s1$ et $s2$ et est appelé score optimal des deux chaînes. Dans le cas de l'exemple ci-dessus, $\mathcal{A}$ représente le seul alignement optimal de $s1$ et $s2$. Mais il est clair que deux chaînes $s1$ et $s2$ peuvent posséder plusieurs alignements optimaux. Un exemple est donné ci-dessous :

Exemple 3.2.7 *Soient les deux chaînes $s1$ et $s2$ définies sur l'alphabet $\Sigma = \{a, b, c\}$ et données par :*

$s1 = abca$

$s2 = acba.$

Supposons que $\mathcal{M}$ soit la matrice identité et $w_\lambda = -\frac{1}{2}$. Alors : $\mathcal{A} = \begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} b \\ c \end{pmatrix}\begin{pmatrix} c \\ b \end{pmatrix}\begin{pmatrix} a \\ a \end{pmatrix}$ et $\mathcal{B} = \begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} \lambda \\ c \end{pmatrix}\begin{pmatrix} b \\ b \end{pmatrix}\begin{pmatrix} c \\ \lambda \end{pmatrix}\begin{pmatrix} a \\ a \end{pmatrix}$ sont deux alignements optimaux (de score 2) de $s1$ et $s2$.

Dans sa forme présente, la mesure de la ressemblance a porté sur deux objets uniquement qui sont comparés directement entre eux. La comparaison d'objets peut se faire d'autres façons. C'est ce que nous allons montrer en commençant par parler des méthodes locales qui vont nous permettre de repérer les objets qui nous intéressent : les mots communs à un ensemble de chaînes.

3.3 Méthodes de comparaison

3.3.1 Comparaison globale ou locale

Dans les mesures de ressemblance que nous avons présentées jusqu'à maintenant, c'est dans leur totalité que nous comparions les objets. Or nous avons montré dans le chapitre 2 que c'est localement surtout que ces objets vont nous intéresser.

La manière classique de procéder pour effectuer des comparaisons locales de chaînes demeure cependant essentiellement la même : par un alignement implicite ou explicite des objets à comparer.

Introduisons donc d'abord une notation concernant l'alignement de deux mots dans une chaîne.

Définition 3.3.1 *Soient* $s1 = a_1...a_n$ *et* $s2 = b_1...b_m$ *deux chaînes et soient* (i, j) *et* (k, l) *deux paires d'entiers tels que* $1 \leq i < j \leq n$, $1 \leq k < l \leq m$ *et* $j - i \leq l - k$. *Ces paires définissent ainsi deux mots* $u1 = a_i...a_j$ *et* $u2 = b_k...b_l$ *de* $s1$ *et* $s2$ *respectivement. Alors l'alignement* $\begin{pmatrix} \overline{a}_1 \\ \overline{b}_1 \end{pmatrix} \begin{pmatrix} \overline{a}_2 \\ \overline{b}_2 \end{pmatrix} ... \begin{pmatrix} \overline{a}_p \\ \overline{b}_p \end{pmatrix}$ *avec* $(j - i + 1) \leq p \leq (j - i + 1) + (l - k + 1)$ *tel que :*

- $\overline{a}_r = a_s$ *ou* $\overline{a}_r = \lambda$ *pour* $1 \leq r \leq p$ *et* $i \leq s \leq j$
- $\overline{b}_r = b_s$ *ou* $\overline{b}_r = \lambda$ *pour* $1 \leq r \leq p$ *et* $k \leq s \leq l$
- $| \overline{a}_i...\overline{a}_{i+p} | = j - i + 1$ *et* $\overline{a}_1...\overline{a}_p = a_i...a_j$
- $| \overline{b}_k...\overline{b}_{k+p} | = l - k + 1$ *et* $\overline{b}_1...\overline{b}_p = b_k...b_l$
- *pour tout* $r \in \{1, ..., p\}$, *il n'est jamais vrai que* $\overline{a}_r = \lambda = \overline{b}_r$

est un alignement des mots $u1$ *et* $u2$.

Exemple 3.3.1 *Soient* $\Sigma = \{a, b, c, d\}$ *et les deux chaînes définies sur cet alphabet données ci-dessous:*

$s1 = abcbdcb$

$s2 = acbbdca$.

Alors $\begin{pmatrix} b \\ b \end{pmatrix} \begin{pmatrix} d \\ d \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix}$ *est un alignement du mot* $u1 = bd$ *de* $s1$ *avec le mot* $u2 = bdc$ *de* $s2$.

Le score de l'alignement des mots $u1$ et $u2$ est défini exactement de la même façon que le score des deux chaînes $s1$ et $s2$ dont ils font partie, et à nouveau ici la mesure de ressemblance entre $u1$ et $u2$ correspond traditionnellement à celle du meilleur alignement possible des deux mots, c'est-à-dire, celui ayant un score optimal. Le score de l'alignement de deux mots peut représenter aussi en fait, comme dans le cas des chaînes, une distance, par exemple celle de Levenshtein.

Exemple 3.3.2 *La distance de Levenshtein séparant les deux mots bd et bdc de l'exemple 3.3.1 est égale à 1.*

Nous allons alors nous intéresser surtout à identifier les mots qui se ressemblent suffisamment dans ces chaînes car ce sont ceux-là que nous allons vouloir définir comme étant similaires. Avec tout ce que nous venons d'établir, cette identification est immédiate :

Définition 3.3.2 *Étant donné deux chaînes* $s1 = a_1...a_n$ *et* $s2 = b_1...b_m$ *et une constante seuil, deux mots* $u1$ *et* $u2$ *de* $s1$ *et* $s2$ *respectivement sont considérés comme étant similaires si* $score\mathcal{A} \geq$ *seuil avec* $\mathcal{A}$ *un meilleur alignement de* $u1$ *et* $u2$ *(ou* $score\mathcal{A} \leq$ *seuil si le score d'un alignement mesure une dissimilarité ou une distance).*

Observons enfin que lorsque nous travaillons avec une mesure de similarité, il est possible d'attribuer une valeur à la ressemblance locale de deux chaînes. Celle-ci est en effet donnée par le plus grand score obtenu parmi tous les alignements (optimaux) possibles de mots dans ces chaînes [Smith and Waterman, 1981].

Définition 3.3.3 *Soient* $s1$ *et* $s2$ *deux chaînes et un système de score ayant pour base sur une mesure de similarité. Soit* $VRL(s1, s2)$ *la valeur de la ressemblance locale de* $s1$ *et* $s2$*. Alors :*

$$VRL(s1, s2) = max\ \{score\mathcal{A} \mid \mathcal{A}\ \textit{un alignement optimal possible de}\ u1\ \textit{et}\ u2\ \textit{pour tous mots}\ u1\ \textit{et}\ u2\ \textit{dans}\ s1\ \textit{et}\ s2\ \textit{respectivement}\}.$$

Bien sûr, une telle valeur n'a vraiment de sens que si certains des scores attribués aux paires de monomères (et *a fortiori* aux paires symbole-trou) sont négatifs.

3.3.2 Comparaison deux à deux ou multiple

Il y a essentiellement deux raisons pour lesquelles comparer plusieurs objets à la fois peut nous permettre de réaliser une analyse plus fine de leur degré de ressemblance. La principale est liée à l'idée de consistance. Considérons un ensemble S de $n > 2$ objets. Même si nous réalisons toutes les comparaisons deux à deux possibles de ces objets, comme chaque comparaison est indépendante, aucune ne peut faire profiter les autres automatiquement des résultats qu'elle obtient. Bien qu'une comparaison multiple, simultanée, de tous les éléments de S ne représente en fait rien d'autre qu'un très grand nombre de comparaisons deux à deux, ces dernières partagent cependant alors ce que l'on peut appeler une 'mémoire associative' en quelque sorte.

Illustrons ce point sur un exemple simple. Supposons que nous sommes en train de comparer trois chaînes $s1$, $s2$, and $s3$, et que, étant donné une certaine définition de ressemblance entre les mots de ces chaînes (par exemple celle basée sur la distance de Hamming), les mots u dans $s1$, v dans $s2$ et w dans $s3$ sont tous similaires entre eux (c'est-à-dire, leurs distances deux à deux sont toutes inférieures à une certaine valeur). Un exemple est donné dans la figure 3.15 : au maximum 2 substitutions sont nécessaires pour passer de l'un quelconque de ces mots à chacun des deux autres. Nous comparons maintenant ces chaînes soit de manière binaire en effectuant toutes les combinaisons 2 à 2 possibles de $s1$, $s2$ et $s3$, soit de manière multiple impliquant les 3 chaînes en même temps.

Dans le premier cas, nous observons que les mots u et v sont similaires, de même que v et w ainsi que u et w. Sans réaliser cependant un travail supplémentaire sur ces résultats préliminaires, il n'est pas possible de noter qu'en fait $\{u, v, w\}$ forme un seul groupe de mots similaires. Une comparaison multiple est capable d'établir ces liens de manière automatique. Elle aussi procède par comparaison de deux objets seulement à chaque fois, mais elle est capable de conserver la trace de l'information que chacune de ces comparaisons deux à deux fournit de manière à pouvoir les récupérer facilement ensuite.

La seconde raison pour laquelle une comparaison multiple est plus intéressante qu'un grand nombre de comparaisons deux à deux peut être appelée l'"amplification de la signification' d'une ressemblance. Un autre exemple illustre ce point dans la figure 3.16 (a). Les similarités entre u et v, v et w, et u et w, sont maintenant plus lointaines (3 substitutions sont nécessaires pour passer de l'un de ces mots à chacun des deux autres au lieu de 2 comme dans l'exemple

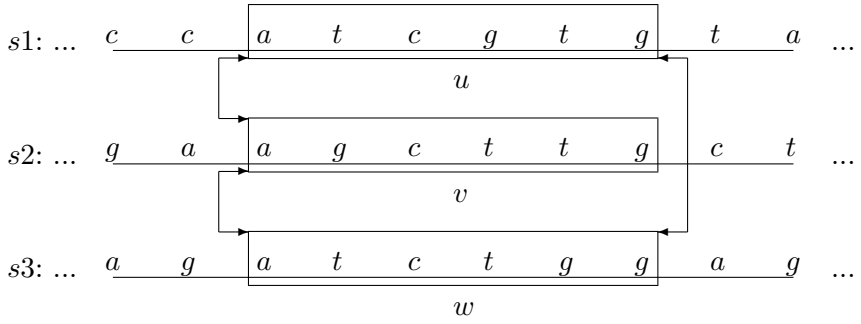


Figure 3.15: Illustration du concept de consistance dans une comparaison multiple.

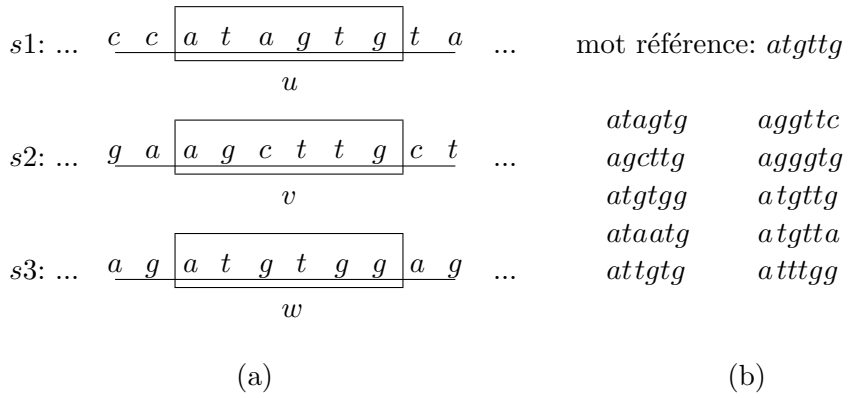


Figure 3.16: Illustration du concept de l'amplification de la signification dans une comparaison multiple.

précédent (figure 3.15)). Ces similarités sont devenues suffisamment faibles en fait pour que nous puissions imaginer qu'une comparaison binaire locale de deux quelconques des chaînes $s1$, $s2$ et $s3$ produise un résultat qui serait considéré comme non-significatif sauf à relâcher tellement les critères de similarité que presque tout finirait par se ressembler. Dans une comparaison multiple, nous obtenons bien sûr que ces trois mots se ressemblent tous entre eux d'une certaine façon. Et dans ce cas, ceci pourrait être vu comme étant significatif car, malgré le fait que la ressemblance de chacune des paires de mots possibles n'est jamais assez forte en elle-même, leur co-existence en est un renforcement. Cet effet devient d'autant plus marqué qu'au lieu d'avoir trois mots faiblement similaires entre eux, nous en avons dix (comme dans la figure 3.16), ou même une centaine. Un exemple concret et très illustratif d'un tel phénomène est donné dans le chapitre consacré aux illustrations.

C'est donc à des comparaisons multiples que nous allons nous attacher dans tout le reste de ce travail. Nous ne parlons des comparaisons deux à deux que dans la mesure où elles servent

de base de départ pour une comparaison multiple. En particulier, le score d'un alignement multiple par exemple, qu'il s'agisse de chaînes ou de mots, est classiquement obtenu à partir de celui des alignements deux à deux le composant.

Bien sûr, réaliser une comparaison multiple est un problème intrinsèquement combinatoire que le concept de relation (et ses cliques maximales) et de couverture permettent toutefois de rendre efficace. Dans ce cas aussi, il est souvent important d'utiliser une méthode indirecte de comparaison et c'est de la distinction existant entre les méthodes directe et indirecte dont nous parlons maintenant.

3.3.3 Comparaison directe ou indirecte

Il y a deux manières de comparer des objets lorsque nous cherchons à en identifier des parties communes. Cette observation est valable quel que soit le nombre d'objets à analyser, mais la distinction devient particulièrement intéressante en termes pratiques lorsqu'il s'agit de comparer plus de deux objets à la fois.

Prenons le cas concret qui nous intéresse ici où ces objets sont des chaînes de caractères et ce que nous cherchons à identifier sont tous les groupes de mots similaires présents dans ces chaînes. La première manière de faire procède par une comparaison directe de ces mots entre eux. S'il y a un nombre total p de mots, il faut alors, si nous voulons être exhaustifs, procéder à au moins 2^p comparaisons. La seconde façon d'agir utilise une approche indirecte. Elle identifie ainsi les groupes de mots similaires en comparant les mots non plus directement entre eux, mais plutôt à des objets qui sont externes aux chaînes. Ces objets n'ont pas besoin d'être des mots, ils peuvent par exemple représenter un produit cartésien d'éléments d'une couverture. Même lorsqu'il s'agit de mots, c'est-à-dire lorsqu'ils sont définis sur le même alphabet que les chaînes, ces objets peuvent ne pas être eux-mêmes présents dans ces chaînes. Dans tous les cas, ils font ainsi fonction de points de références contre lesquels tous les objets internes (les mots dans les chaînes) sont comparés. Un ensemble de mots similaires est alors un ensemble de mots tels qu'il existe un de ces points de référence avec lequel tous les mots de l'ensemble se trouvent dans une certaine relation de similarité (voir la figure 3.17). Le nombre de comparaisons à effectuer pour le même total p de mots varie suivant la relation adoptée, nous reviendrons là-dessus plus tard, mais notons seulement pour l'instant qu'il sera en général considérablement plus petit que 2^p. Dans certains cas, il peut être proportionnel à p. Cette idée d'utiliser un objet externe pour réaliser une comparaison multiple de mots d'une chaîne a été, à notre connaissance, initialement introduite par Waterman [Galas *et al.*, 1985] [Waterman, 1984b] [Waterman, 1986] [Waterman, 1989] [Waterman, 1990] [Waterman *et al.*, 1984].

Chacun de ces objets externes représente également d'une certaine façon un 'résumé' d'un groupe de mots similaires. De tels résumés sont obtenus même lorsque les comparaisons sont effectuées de manière directe mais il est important de distinguer les deux. Dans ce second cas, le résumé des mots similaires est toujours obtenu après que les mots aient été identifiés. Dans le premier cas où l'on procède à des comparaisons indirectes, le résumé est le plus souvent établi en même temps que les groupes de mots sont trouvés. Lorsque cela est ainsi, ces objets externes peuvent être vus comme des constructeurs de ces groupes et c'est ce type d'objet qui est utilisé en ce qui concerne notre propre travail.

Dans les deux sections qui suivent, nous allons affiner les notions générales que nous venons d'introduire et montrer aussi que la différence entre comparaisons directe et indirecte est en fait souvent mince. Ceci va nous conduire à présenter jusqu'à la fin de ce chapitre une série de définitions de similarité entre mots dont nous nous servirons par la suite pour identifier les groupes de mots qui se ressemblent suffisamment entre eux dans l'espoir que ces groupes

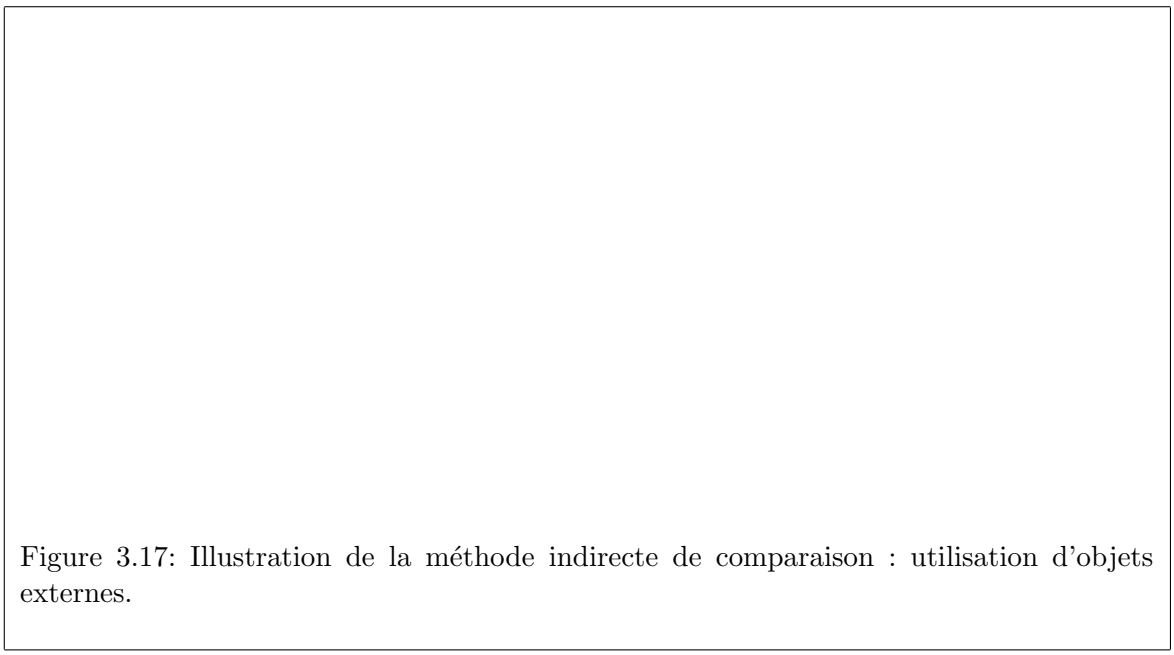


Figure 3.17: Illustration de la méthode indirecte de comparaison : utilisation d'objets externes.

reflètent certaines caractéristiques fonctionnelles, structurales ou évolutives des molécules comparées.

Chaque groupe de mots similaires entre eux d'une certaine façon constitue ce qui est appelé un bloc [Henikoff, 1994] [Henikoff and Henikoff, 1991] [Posfai *et al.*, 1989] (voir la figure 3.18).

Notation 3.3.1 *Étant donné un ensemble de chaînes, nous appelons bloc un groupe de mots dans ces chaînes similaires suivant une certaine mesure de ressemblance.*

Comme nous l'avons mentionné dans le chapitre 2, un bloc peut comporter des mots présents dans chacune des chaînes que l'on analyse, ou seulement dans un sous-ensemble de celles-ci. Nous allons donc introduire là où cela est possible une contrainte supplémentaire, que nous appelons contrainte de quorum. Cette contrainte va indiquer le nombre minimum de chaînes de l'ensemble où au moins un mot d'un bloc de similarité doit apparaître pour que ce bloc soit considéré comme intéressant. Cette contrainte est essentielle car elle permet de traiter le cas où les caractéristiques communes que nous voulons déterminer dans l'ensemble des chaînes ne sont pas en fait observées sur toutes celles-ci. Ce cas, qui correspond à un ensemble de données comportant du bruit, est très fréquemment rencontré en pratique. Observons que cette contrainte intervient également lorsque nous travaillons avec une seule chaîne : elle indique alors le nombre minimum de fois où un mot doit apparaître répété dans la chaîne pour mériter notre attention. Nous en donnons ici une définition formelle.

Définition 3.3.4 *Étant donné un ensemble $\mathcal{S}$ de chaînes définies sur Σ^* et un bloc B de mots similaires dans $\mathcal{S}$, on appelle :*

- *le quorum en mots q_m de B la cardinalité de B :*

$$q_m(B) = Card(B);$$

- *le quorum en chaîne q_c de B le nombre de chaînes (distinctes) de $\mathcal{S}$ dans lesquelles au moins un mot de B est présent :*

$$q_c(B) = \sum_{s \in \mathcal{S}} \delta_0(B, s)$$

$$\text{où } \delta_0(B, s) = \begin{cases} 1 \text{ si } \exists u \in B \text{ tel que } u \text{ est un mot dans } s \\ 0 \text{ sinon} \end{cases}$$

Comme nous allons nous soucier surtout du quorum en chaîne, dorénavant lorsque nous parlerons de quorum, c'est donc au quorum q_c que nous ferons référence. Lorsque cela ne sera pas le cas, nous l'indiquerons de manière explicite.

Notation 3.3.2 *Nous appelons quorum d'un bloc B de mots similaires d'un ensemble de chaînes, le quorum en chaîne q_c de B.*

Nous disons alors que :

Définition 3.3.5 *Un bloc B de mots similaires d'un ensemble N de chaînes définies sur Σ^* satisfait la contrainte de quorum q si $q_c(B) \geq q$.*

Finalement, observons qu'il peut être important dans certaines situations de repérer des mots qui sont non plus communs à un nombre suffisamment grand de chaînes d'un ensemble, mais observés sur un nombre 'anormalement' petit de celles-ci. En d'autres termes, il peut être important de repérer les mots qui sont rares. Cette idée a été énoncée par [Blaisdell *et al.*, 1993] et [Waterman, 1989] en particulier et est résumée de manière très suggestive par Watanabe [Watanabe, 1985] : " *The holes in Emmenthal cheese (or lotus root) certainly can be considered as an important constituent part of the object. This reinterpretation of absence as presence may be compared with the interpretation of a 'bubble' in the sea of negative electrons in negative energy states as a positive electron*".

Nous pouvons ainsi vouloir définir le concept d'une contrainte maximale de quorum q en disant qu'un bloc B de mots similaires d'un ensemble N de chaînes définies sur Σ^* satisfait cette contrainte si $q_c(B) \leq q$. La contrainte de quorum de la définition 3.3.5 est dans ce cas une contrainte que nous appelons minimale. Les deux notions ne sont toutefois pas symétriques. En effet, nous avons :

- $q_c(B) \geq q \Longrightarrow q_m(B) \geq q$
  mais
- $q_c(B) \leq q \not\Longrightarrow q_m(B) \leq q$.

Un bloc peut ainsi satisfaire une contrainte maximale de quorum et cependant être présent un nombre 'anormalement' grand de fois dans une des chaînes de l'ensemble. Bien sûr, un bloc peut satisfaire une contrainte minimale de quorum et être totalement absent d'une des chaînes, mais biologiquement les deux phénomènes ne seront sans doute pas considérés comme étant également significatifs. Traiter les mots rares est aussi techniquement plus difficile et bien que représentant un problème très intéressant, nous considèrons dans ce travail uniquement les blocs satisfaisant une contrainte minimale de quorum. Le terme plus général de contrainte de quorum fera ainsi référence à cette dernière.

Nous allons commencer par décrire les mesures directes de ressemblance, et d'abord par une approche qui demeure globale. Celle-ci est importante car c'est avec elle que nous montrons comment passer de la valeur d'un alignement deux à deux donnée précédemment à celle d'un alignement multiple, puis à celle d'un bloc. La première définition de similarité entre mots que nous donnons a ainsi toujours pour base l'optimisation d'une fonction de score.

Figure 3.18: Exemple d'un bloc de similarité (2 substitutions au plus autorisées entre chaque paire de mots du bloc).

3.4 Mesures directes de la ressemblance

3.4.1 Approche globale : optimisation d'une fonction de score

3.4.1.1 Deux façons d'aligner plusieurs chaînes

L'approche globale et directe pour mesurer la ressemblance entre deux objets correspond traditionnellement à celle présentée dans la section 3.2.4.2 [Needleman and Wunsch, 1970] [Sellers, 1974] [Wagner and Fischer, 1974]. La mesure de ressemblance de deux chaînes $s1$ et $s2$ est ainsi donnée par $VR(s1,s2)$.

Le problème est que, comme nous l'avons vu, nous allons vouloir comparer toujours plus de deux objets à la fois. En général, il s'agit de comparer plusieurs mots simultanément, mais nous pouvons aussi vouloir traiter des chaînes entières.

La mesure de la ressemblance globale de plus de deux chaînes que l'on compare directement entre elles est une extension de la mesure de ressemblance de deux de ces objets seulement. À la base de cette mesure multiple se trouve donc toujours un alignement comme dans le cas du deux à deux. C'est dans la façon d'obtenir cet alignement que des différences surgissent et il existe ainsi deux manières principales de procéder. Nous n'allons pas nous soucier pour l'instant de méthodes, celles-ci sont montrées rapidement dans le chapitre suivant, mais seulement de l'idée qui se cache derrière chacune des deux approches. Pour cela, revenons un peu à la biologie et rappelons que ces chaînes représentent des objets ayant une histoire évolutive éventuellement commune. Les deux méthodes d'alignement multiple varient ainsi suivant qu'il est ou non tenu compte de cette histoire. Dans le second cas, comme aucune supposition *a priori* n'est faite des liens existant possiblement entre les objets, ceux-ci sont tous comparés entre eux. Dans le premier, on essaye de reconstituer l'histoire évolutive des objets, au préalable ou en même temps que l'alignement, construisant ainsi ce que l'on appelle un arbre phylogénétique, et chaque objet n'est comparé qu'avec certains des autres objets de l'ensemble, ceux auxquels il est lié par une branche de l'arbre. En fait, cette approche s'assimile d'une certaine façon aux méthodes indirectes dans la mesure où les objets initiaux sont en réalité souvent comparés à des objets ancestraux plutôt que directement ou, du moins, que ces objets ancestraux sont établis en même temps que l'alignement se réalise.

Dans les deux cas cependant, l'alignement obtenu à la fin présente la forme d'un tableau, comme pour deux chaînes, seulement avec plus de dimensions. Établissons donc d'abord une

notation pour un tel alignement.

Définition 3.4.1 *Soient* $s1 = s1_1...s1_{n_1}$, $s2 = s2_1...s2_{n_2}$, ..., $sN = sN_1...sN_{n_N}$ *N chaînes définies sur* Σ^*. *Alors* $\begin{pmatrix} \overline{s1}_1 \\ \overline{s2}_1 \\ ... \\ \overline{sN}_1 \end{pmatrix} \begin{pmatrix} \overline{s1}_2 \\ \overline{s2}_2 \\ ... \\ \overline{sN}_2 \end{pmatrix} ... \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ ... \\ \overline{sN}_p \end{pmatrix}$ *avec* $min\{n_1, n_2, ..., n_N\} \leq p \leq n_1 + n_2 + ... + n_N$ *tel que :*

- $\overline{si}_j = si_k$ *ou* $\overline{si}_j = \lambda$ *pour* $1 \leq k \leq n_i$ *et* $1 \leq j \leq p$
- $| \overline{si}_1...\overline{si}_p | = n_i$ *et* $\overline{si}_1...\overline{si}_p = si_1...si_{n_i}$
- *pour tout* $j \in \{0, ..., p\}$, *il n'est jamais vrai que pour tout* $i \in \{1, ..., N\}$, $\overline{si}_j = \lambda$ *(aucune colonne de l'alignement n'est composée que de trous)*

représente un alignement des chaînes $s1$, $s2$, ..., sN.

Exemple 3.4.1 *Soient* $\Sigma = \{a, b, c, d, e\}$ *et les chaînes* $s1$, $s2$, $s3$ *et* $s4$ *données par :*
$s1 = acdeceb$
$s2 = acaebeb$
$s3 = cdbeb$
$s4 = bcdbc$.
Alors $\mathcal{A} = \begin{pmatrix} a \\ a \\ \lambda \\ b \end{pmatrix} \begin{pmatrix} c \\ c \\ c \\ c \end{pmatrix} \begin{pmatrix} d \\ a \\ d \\ d \end{pmatrix} \begin{pmatrix} e \\ e \\ \lambda \\ \lambda \end{pmatrix} \begin{pmatrix} c \\ b \\ b \\ b \end{pmatrix} \begin{pmatrix} e \\ e \\ e \\ e \end{pmatrix} \begin{pmatrix} b \\ b \\ b \\ \lambda \end{pmatrix}$ *est un alignement multiple de* $s1$, $s2$, $s3$ *et* $s4$.

La question est maintenant : quel score attribuer à cet alignement? C'est dans cette attribution des scores que va justement apparaître la différence entre un alignement obtenu par comparaison de toutes les chaînes entre elles et un alignement obtenu à partir d'un arbre phylogénétique. Voyons d'abord le premier.

3.4.1.2 Les scores SP

Dans le cas où toutes les chaînes d'un ensemble sont comparées entre elles, il est naturel de considérer que le score de l'alignement multiple est la somme des scores de tous les alignements deux à deux qu'il induit, en d'autres termes, ceux qui sont la projection de cet alignement sur chacune des paires de chaînes de l'ensemble (voir la figure 3.19).

Dans ce cas, le score d'une colonne de l'alignement est la somme des scores de toutes les paires de symboles ou paires symbole-trou de cette colonne. Ces scores sont appelés des scores SP, de l'anglais 'Sum of the Pairs' [Altschul and Lipman, 1989].

Plus formellement, nous pouvons établir une définition de ce score qui ressemble à celle du cas deux à deux, en particulier elle aussi est récursive.

Définition 3.4.2 *Soit* $\mathcal{A} = \begin{pmatrix} \overline{s1}_1 \\ \overline{s2}_1 \\ ... \\ \overline{sN}_1 \end{pmatrix} \begin{pmatrix} \overline{s1}_2 \\ \overline{s2}_2 \\ ... \\ \overline{sN}_2 \end{pmatrix} ... \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ ... \\ \overline{sN}_p \end{pmatrix}$ *avec* $min\{n_1, n_2, ..., n_N\} \leq p \leq n_1 + n_2 + ... + n_N$ *un alignement des chaînes* $s1 = s1_1...s1_{n_1}$, $s2 = s2_1...s2_{n_2}$, ..., $sN = sN_1...sN_{n_N}$. *Alors :*

Figure 3.19: Alignement multiple utilisant des scores SP et alignement deux à deux induit de cet alignement multiple.

$$score\mathcal{A} = score\mathcal{A}' + score \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ ... \\ \overline{sN}_p \end{pmatrix}$$

où :

$$\mathcal{A}' = \begin{pmatrix} \overline{s1}_1 \\ \overline{s2}_1 \\ ... \\ \overline{sN}_1 \end{pmatrix} \begin{pmatrix} \overline{s1}_2 \\ \overline{s2}_2 \\ ... \\ \overline{sN}_2 \end{pmatrix} ... \begin{pmatrix} \overline{s1}_{p-1} \\ \overline{s2}_{p-1} \\ ... \\ \overline{sN}_{p-1} \end{pmatrix}$$

et

$$score \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ ... \\ \overline{sN}_p \end{pmatrix} = \sum_{1 \leq i < j \leq N} score \begin{pmatrix} \overline{si}_p \\ \overline{sj}_p \end{pmatrix}$$

avec $score \begin{pmatrix} \overline{si}_p \\ \overline{sj}_p \end{pmatrix}$ *donné par la définition 3.2.19.*

Exemple 3.4.2 *Si nous travaillons avec une mesure de similarité et une relation d'identité entre les symboles de l'alphabet, alors le score de l'alignement des trois chaînes de la figure 3.19 est égal à* 6.

La pondération affectée aux trous dans un alignement multiple est celle qu'utilise Gotoh [Gotoh, 1982]. Elle correspond à celle que nous avons décidée d'adopter : chaque trou dans un alignement reçoit un poids constant. Comme pour l'attribution d'un score à un trou dans le cas de deux chaînes, d'autres fonctions ont été élaborées, notamment par Altschul [Altschul and Lipman, 1989], Fredman [Fredman, 1984] et Murata [Murata *et al.*, 1985]. Altschul en particulier n'affecte un score qu'à l'ouverture d'un trou dans l'alignement induit sur chaque paire, pas à son prolongement. La différence entre le score de Gotoh et celui d'Altschul est montré dans la figure 3.20 à l'aide d'exemples. Il est clair que les coûts d'Altschul sont plus complexes à calculer, ils impliquent un retour en arrière constant dans l'alignement pour vérifier où a commencé un trou.

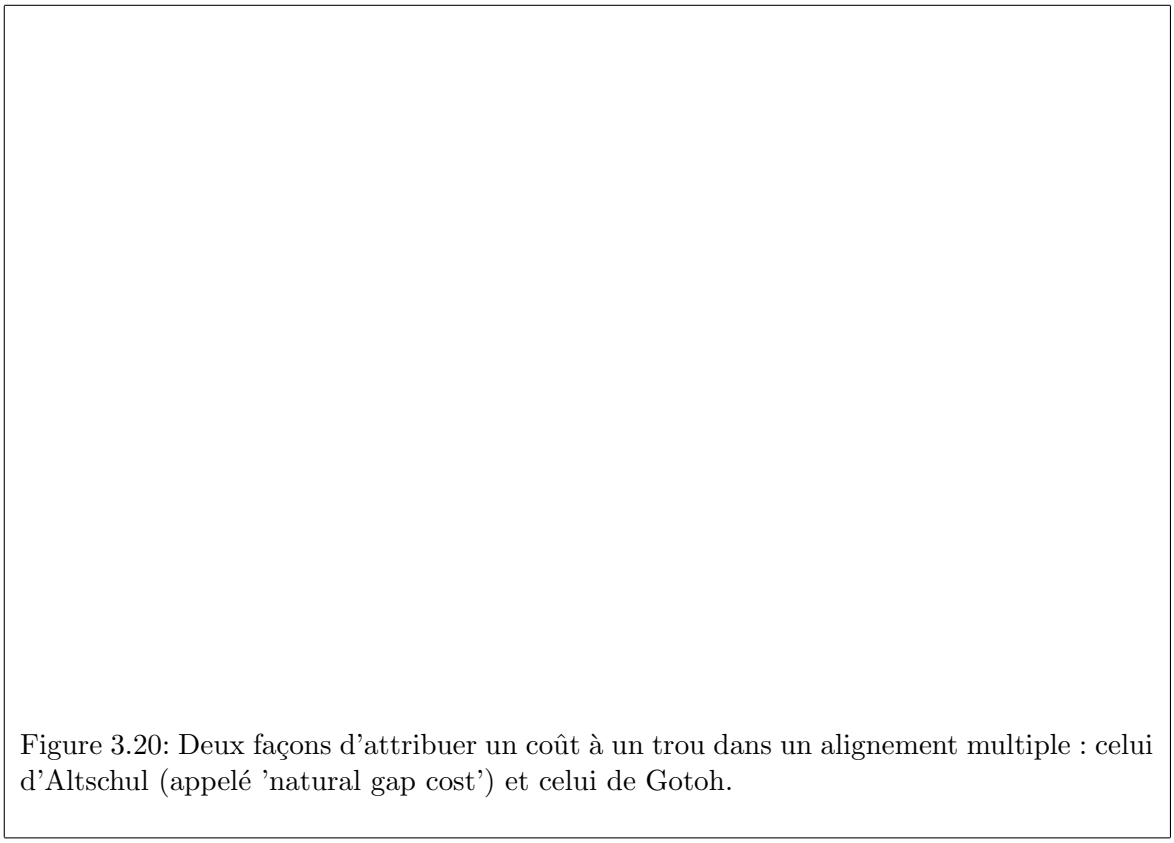


Figure 3.20: Deux façons d'attribuer un coût à un trou dans un alignement multiple : celui d'Altschul (appelé 'natural gap cost') et celui de Gotoh.

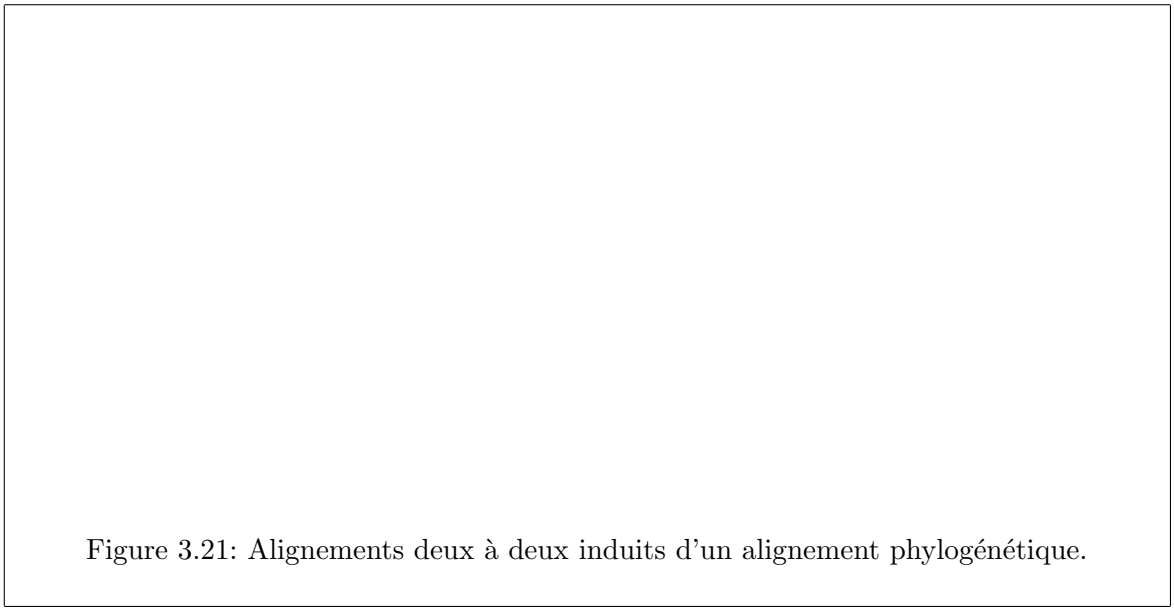


Figure 3.21: Alignements deux à deux induits d'un alignement phylogénétique.

3.4.1.3 Scores obtenus en utilisant un arbre

Une des motivations pour l'étude d'objets qui ont évolué au cours du temps à partir d'un certain nombre d'ancêtres communs est d'essayer justement d'établir leur histoire et les liens existant entre ces objets. Ce processus peut fonctionner dans le sens inverse cependant. En effet, si l'histoire évolutive d'un ensemble taxonomique a pu être bien établie à partir de données fossiles ou morphologiques, cette histoire peut être utilisée pour éclairer l'évolution au niveau moléculaire des chaînes biologiques et donc pour comparer et aligner des macromolécules. Une première formalisation de cette approche, appelée alignement multiple à partir d'un arbre, est due principalement à Sankoff [Sankoff *et al.*, 1973] [Sankoff, 1975] [Sankoff *et al.*, 1976]. On appelle cela aussi réaliser un alignement multiple phylogénétique.

Habituellement, les chaînes de l'ensemble à aligner constituent les feuilles d'un tel arbre, et des chaînes additionnelles, correspondant à ce que l'on pourrait appeler des chaînes ancestrales, sont placées aux nœuds internes de l'arbre. Ces chaînes ancestrales sont construites pendant l'alignement et sont ajoutées à celui-ci. L'alignement multiple final présente ainsi la forme de deux tableaux placés l'un en dessous de l'autre, dont seul celui du dessus nous intéresse. Celui-ci en effet concerne l'alignement des chaînes de départ, et celui du dessous l'alignement des chaînes ancestrales. Le score de cet alignement des chaînes de départ plus les chaînes ancestrales est alors la somme des scores de tous les alignements deux à deux associés à chaque branche de l'arbre et qui sont induits par l'alignement multiple (voir la figure 3.21) [Sankoff, 1975]. Ces alignements deux à deux portent donc toujours sur une chaîne de départ et une chaîne ancestrale.

Formellement, nous avons :

Définition 3.4.3 *Soit* $\mathcal{A} = \begin{pmatrix} \overline{s1}_1 \\ \overline{s2}_1 \\ ... \\ \overline{sN}_1 \end{pmatrix} \begin{pmatrix} \overline{s1}_2 \\ \overline{s2}_2 \\ ... \\ \overline{sN}_2 \end{pmatrix} ... \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ ... \\ \overline{sN}_p \end{pmatrix}$ *avec* $min\{n_1, n_2, ..., n_N\} \leq p \leq n_1 + n_2 + ... + n_N$ *un alignement des chaînes* $s1 = s1_1...s1_{n_1}$, $s2 = s2_1...s2_{n_2}$, ..., $sN =$

$sN_1 ... sN_{n_N}$ obtenu à partir d'un arbre phylogénétique ayant M nœuds internes, et donc M chaînes ancestrales a1,a2,...,aM qui ont été reconstruites en même temps que l'alignement et qui sont de longueurs $m_1, m_2, ..., m_M$ respectivement. Alors :

$$score\mathcal{A} = score\mathcal{A}' + score \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ ... \\ \overline{sN}_p \end{pmatrix}$$

où :

$$\mathcal{A}' = \begin{pmatrix} \overline{s1}_1 \\ \overline{s2}_1 \\ ... \\ \overline{sN}_1 \end{pmatrix} \begin{pmatrix} \overline{s1}_2 \\ \overline{s2}_2 \\ ... \\ \overline{sN}_2 \end{pmatrix} ... \begin{pmatrix} \overline{s1}_{p-1} \\ \overline{s2}_{p-1} \\ ... \\ \overline{sN}_{p-1} \end{pmatrix}$$

et

$$score \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ ... \\ \overline{sN}_p \end{pmatrix} = \sum_{\substack{1 \leq i \leq N \\ 1 \leq j \leq M \\ aj \text{ chaîne ancestrale} \\ \text{auquelle } si \text{ est liée}}} score \begin{pmatrix} \overline{si}_p \\ aj_{m_j} \end{pmatrix}$$

avec $score \begin{pmatrix} \overline{si}_p \\ aj_{m_j} \end{pmatrix}$ *donné par la définition 3.2.19 sachant que* $aj_{m_j} \in \Sigma$.

Exemple 3.4.3 *Si nous travaillons avec une mesure de distance et une relation d'identité entre les symboles de l'alphabet, alors le score de l'alignement des trois chaînes de la figure 3.21 est égal à* 6.

Comme nous venons de le dire, l'arbre phylogénétique est donné au départ, il est obtenu en fait à partir d'autres mesures. Mais en réalité, il est possible de procéder dans les deux directions à la fois, c'est-à-dire de faire usage d'un arbre pour obtenir un meilleur alignement des molécules comparées, et utiliser l'alignement progressivement obtenu pour construire l'arbre. Dans ce cas au départ il n'y a rien : ni alignement, ni arbre. Lorsque l'arbre est connu au départ, nous connaissons le nombre d'ancêtres, donc le nombre de nœuds internes de l'arbre ainsi que sa forme, et ce qu'il faut reconstituer ce sont alors uniquement les chaînes ancestrales. Lorsque l'arbre n'est pas fourni initialement, et que sa forme et le nombre d'ancêtres ne sont donc pas connus, c'est tout cela plus les chaînes ancestrales qu'il faut découvrir. Nous indiquons dans la section suivante quel critère est habituellement adopté pour opérer ces reconstructions.

3.4.1.4 Mesure de la ressemblance : optimisation d'une fonction de score

Quel que soit le score utilisé, qu'il s'agisse d'un score SP ou de celui obtenu par construction d'un arbre phylogénétique, la mesure de ressemblance d'un ensemble de chaînes est habituellement donnée, comme dans le cas du deux à deux, par le score du meilleur alignement multiple qu'il est possible de réaliser avec ces chaînes.

Définition 3.4.4 *Soient N chaînes s1,s2,...,sN et soit $VRM(s1, s2, ...sN)$ une valeur de la similarité multiple (resp. dissimilarité) de s1,s2,...,sN. Alors :*

$$VRM(s1, s2, ..., sN) = max \text{ (resp. } min) \{score\mathcal{A} \mid \mathcal{A} \text{ un alignement de } s1,s2,...,sN\}.$$

Dans le cas où l'alignement est déterminé à partir d'un arbre, il est usuel de travailler avec des distances et c'est donc le score minimal qui va nous intéresser. Il faut observer alors que la minimisation de la fonction de récurrence présentée dans la section 3.4.1.3 implique en fait une double minimisation du fait que les chaînes ancestrales ne sont pas connues au départ. Ainsi nous avons :

$$\min(score\mathcal{A}) = \min(score\mathcal{A}' + \min(score \begin{pmatrix} \overline{s1}_p \\ \overline{s2}_p \\ \dots \\ \overline{sN}_p \end{pmatrix})).$$

C'est cette double minimisation, l'externe et l'interne, qui constitue le critère utilisé pour reconstituer les chaînes ancestrales d'un arbre lorsque celui-ci est connu à l'avance. Ce critère est appelé critère de parsimonie. Lorsque l'arbre n'est pas connu, pour évaluer la mesure de ressemblance d'un ensemble d'objets en tenant compte de leur histoire évolutive il faut alors encore déterminer, parmi tous les arbres possibles, celui qui obtient un score minimal sachant que les chaînes de départ sont toujours placées aux feuilles. La combinatoire explosive d'une telle approche fait que des algorithmes exacts pour mesurer la ressemblance de plusieurs chaînes à partir d'un arbre n'existent que lorsque celui-ci est connu à l'avance.

3.4.1.5 Deux cas particuliers de score multiple : les étoiles centrales et les arbres étoiles

Les scores SP présentés dans la section précédente sont souvent coûteux à calculer mais leur principal inconvénient est qu'il n'est pas vraiment intuitif de comprendre quel sens ils peuvent avoir en biologie. Que signifie en effet attribuer comme score à un alignement (c'est-à-dire en fait établir comme mesure de sa qualité), la somme de tous les scores de paires de chaînes de l'ensemble puis à considérer que le meilleur alignement est celui dont le score ainsi obtenu est le plus élevé (ou le plus petit si l'on travaille avec des distances)? En quoi cette mesure peut-elle refléter la vraie histoire évolutive des chaînes puisqu'elle agit comme si chacune de celles-ci pouvait être un ancêtre de toutes les autres alors que la plupart n'ont pas de lien évolutif direct entre elles et ne sont en relation que parce qu'elles sont issues d'un ancêtre commun (qui peut ne même pas être présent dans l'ensemble). Sans doute une meilleure mesure d'une telle relation est celle qui s'appuie sur le concept d'étoile centrale (en anglais 'Center Star') introduite par Gusfield [Gusfield, 1993]. Gusfield lui-même considère cette mesure comme une approximation des scores SP et l'a donc établie pour des raisons pragmatiques. Nous la voyons plutôt comme une mesure intéressante en elle-même. Elle consiste à définir le score d'un alignement multiple de la façon suivante :

Définition 3.4.5 *Soient s1,s2,...,sN des chaînes définies sur Σ^*. Le score de similarité (resp. dissimilarité) d'un alignement EC ('Étoile Centrale') de s1,s2,...,sN est égal à :*

$$score(EC) = max\ (resp.\ min)\ \{\sum_{j=1}^{N} score(si, sj) \mid i \in \{1, ..., N\}\}$$

où $score(si, sj)$ est le score d'un alignement optimal des chaînes si et sj. C'est donc le maximum (resp. minimum) de la somme de tous les scores optimaux deux à deux par rapport à si, pour toute chaîne si dans l'ensemble de départ. La chaîne si pour laquelle ce maximum (ou minimum) est atteint est appelé l'étoile centrale. C'est la chaîne (étoile) qui se trouve au centre de toutes les autres chaînes (étoiles).

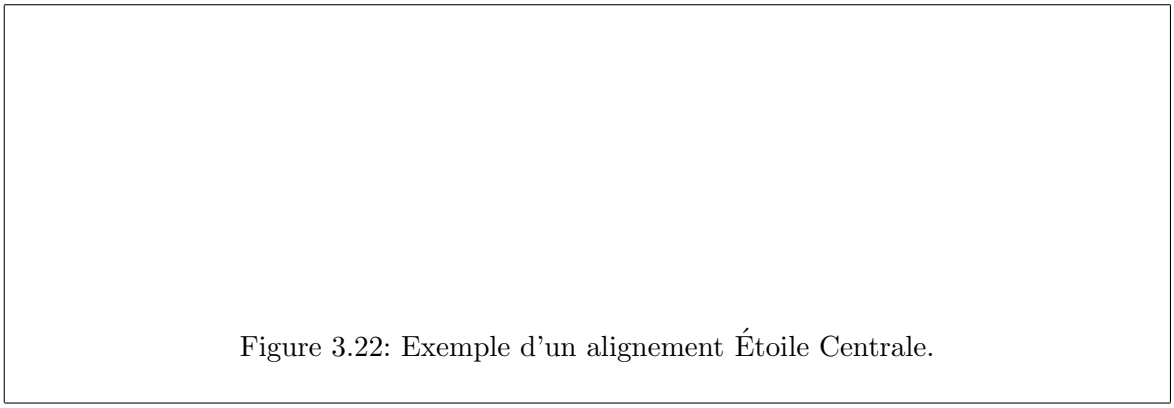


Figure 3.22: Exemple d'un alignement Étoile Centrale.

Exemple 3.4.4 *Si nous travaillons avec une mesure de distance et une relation d'identité entre les symboles de l'alphabet, alors le score de l'alignement des cinq chaînes de la figure 3.22 est égal à* 4.

Une approximation du même genre peut être considérée également lorsque le système de score utilisé s'appuie sur un arbre phylogénétique. Dans ce cas, l'arbre est défini comme étant une étoile : il ne comporte donc qu'un seul nœud interne [Altschul and Lipman, 1989] et les chaînes à comparer sont alors supposées être les descendants directs d'un unique ancêtre commun. Si en termes de phylogénie, cette hypothèse peut paraître absurde en pratique, l'idée d'arbre étoile est au fond tout à fait semblable à celle d'étoile centrale à la seule différence que dans le cas de l'arbre, l'étoile centrale n'a plus besoin d'être une des chaînes de l'ensemble de départ. C'est un concept qu'il est tout à fait intéressant d'explorer dans le cadre des méthodes indirectes de comparaison et que nous reverrons alors sous le nom de centre de gravité ou de mot de Steiner.

3.4.2 Résumé des mesures directes et globales de la ressemblance

Un résumé des mesures directes et globales de la ressemblance est donné dans le tableau ci-dessous.

Mesures Directes et Globales de la Ressemblance	
Caractéristiques principales	Référence aux définitions
optimisation d'une fonction de score scores SP	définition 3.4.4 et définition 3.4.2
optimisation d'une fonction de score scores ayant pour base un arbre phylogénétique	définition 3.4.4 et définition 3.4.3
optimisation d'une fonction de score scores Étoile Centrale	définition 3.4.5

3.4.3 Approche locale

3.4.3.1 Optimisation d'une fonction de score

L'approche locale et directe pour mesurer la ressemblance entre plusieurs objets simultanément commence par une application des méthodes traditionnelles globales que nous avons vues un peu plus haut et fait donc appel encore une fois à l'optimisation d'une fonction de score d'un alignement. Dans ce cas cependant, le score attribué à un alignement de mots dans les chaînes que l'on compare est toujours un score SP et jamais un score obtenu à partir d'un arbre phylogénétique. *A priori*, il n'y a pas de raison pour que les connaissances apportées par un tel arbre ne puissent être utilisées pour comparer des mots, mais la formalisation d'un tel problème devrait sans doute en être différente que celle introduite dans le cas de chaînes entières.

Il suit donc une première définition de similarité entre mots donnée par :

Définition 3.4.6 *Étant donné N chaînes $s1$,$s2$,...,sN et une constante seuil, les mots $u1$,$u2$,..., uN de $s1$,$s2$,...,sN respectivement sont considérés comme étant similaires si score$\mathcal{A} \geq$ seuil avec $\mathcal{A}$ un meilleur alignement multiple de $u1$,$u2$,...,uN (ou score$\mathcal{A} \leq$ seuil si le score d'un alignement mesure une dissimilarité ou une distance).*

À nouveau ici, si nous travaillons avec une mesure de similarité, il est possible d'attribuer une valeur à la ressemblance locale d'un ensemble de chaînes de la façon suivante :

Définition 3.4.7 *Soient N chaînes $s1$,$s2$,...,sN et soit $VRML(s1, s2, ...sN)$ la valeur de la ressemblance multiple locale de $s1$,$s2$,...,sN. Alors :*

$$VRML(s1, s2, ..., sN) = max\ \{score\mathcal{A} \mid \mathcal{A}\ \textit{un alignement de}\ u1,u2,...,uN\ \textit{pour tous mots}\ u1,u2,...,uN\ \textit{dans}\ s1,s2,...,sN\ \textit{respectivement}\}.$$

Encore une fois, cette valeur n'a de sens que si les scores de certaines des paires de monomères sont négatifs. Un groupe de mots $u1$,$u2$,...,uN dont l'alignement produit ce score maximal est alors un bloc des mots les plus similaires qu'il soit possible de trouver dans chacune des chaînes.

Sans entrer pour l'instant dans des considérations de complexité algorithmique sur lesquelles nous reviendrons dans le chapitre suivant, nous pouvons déjà observer que cette définition de similarité entre mots est en pratique la plus coûteuse de toutes avec laquelle travailler, et cela même lorsque les trous ne sont pas permis. Observons aussi que cette définition suppose implicitement que nous fixons la contrainte de quorum q (voir la définition 3.3.5) à une valeur qui correspond en pourcentage à 100% : un bloc de similarité doit avoir au moins une occurrence, c'est-à-dire au moins un mot présent dans chacune des chaînes de l'ensemble. Or nous avons vu qu'il est important de pouvoir travailler en présence de 'bruit', à savoir avec des valeurs variables de quorum. Cela est bien sûr possible mais la définition donnée ici est particulièrement mal adaptée pour traiter de manière algorithmiquement efficace ce genre de situation. Nous en donnerons la raison précise dans le chapitre suivant, mais intuitivement nous pouvons déjà voir que l'introduction d'une telle contrainte augmente la combinatoire du problème. C'est ce qui a motivé l'élaboration d'autres définitions de similarité entre mots. Celle que nous présentons maintenant ne traite que le cas des blocs de mots similaires dont l'alignement ne comporte pas de trous et seulement des substitutions 'conservatives'. Il s'agit en effet d'une extension directe de la relation entre monomères présentée dans la section 3.2.3.2.1 et les substitutions dites conservatives sont celles impliquant des éléments en relation par R.

Soit $\Sigma = \{I, L, M, P, C, W\}$ et soient $\{\{I, L\}, \{L, M\}, \{P\}, \{C\}, \{W\}\}$ les cliques maximales d'une relation R sur Σ.
Soient :

$s1 = LLP$

$s2 = LIC$

$s3 = MLIW$

et $s = s1s2s3 = LLPLICMLIW$.
Alors {1,4,8} et {1,4,7,8} sont des cliques de R_2 mais comme {1,4,8} est inclus dans {1,4,7,8}, ce dernier ensemble est l'unique clique maximale de R_2.

Figure 3.23: Exemple de clique maximale C_k d'une relation R_k établie sur l'alphabet des acides aminés.

3.4.3.2 Recherches combinatoires exactes

Une relation R de similarité entre monomères et le concept de cliques maximales tels que ceux-ci ont été vus dans la section 3.2.3.2 peuvent être étendus de façon très naturelle aux mots, que ceux-ci soient lexicaux 3.2.3.2.1 ou structuraux 3.2.3.2.2. Comme ceux qui nous intéressent sont présents dans un ensemble de chaînes desquelles nous voulons extraire les traits communs, nous avons indiqué (définition 3.1.2) que chacun d'eux peut être identifié de manière non ambiguë par sa position dans ces chaînes. Il est alors traditionnel d'étendre la relation entre monomères en tant que symboles ou en tant qu'objets 3D à une relation entre positions dans les chaînes sachant que celles-ci font référence aux mots. Pour faciliter la notation, nous considérons aussi que les chaînes sont en fait concaténées en une seule, notée s (nous avons donc $s = s1s2...sN$), et la relation de similarité entre mots porte finalement sur leurs indices dans s.

Définition 3.4.8 *Soit une chaîne $s \in \Sigma^n$ et i, j deux positions différentes dans s (c'est-à-dire, deux indices) tels que $1 \leq i, j \leq n$, alors :*

$$i \; R_1 \; j \Leftrightarrow s_i \; R \; s_j.$$

Soient maintenant $k \geq 1$ un entier et i, j deux positions différentes dans s telles que $i, j \leq n - k + 1$, alors :

$$i \; R_k \; j \Leftrightarrow (i + p) \; R_1 \; (j + p) \text{ pour tout } p \in \{0, ..., k - 1\}.$$

Nous disons de i et j vérifiant la condition ci-dessus qu'ils sont en k-relation.

Définition 3.4.9 *Étant donné une chaîne $s \in \Sigma^n$, un ensemble C_k de positions dans s est une clique de la relation R_k si pour tout $i, j \in C_k$, $i \; R_k \; j$. C_k est dit maximal si pour tout $l \in [1..n] \setminus C_k$, $C_k \bigcup \{l\}$ n'est plus une clique.*

Exemple 3.4.5 *Voir la figure 3.23.*

Les cliques maximales d'une relation R_k fournissent alors une façon différente de définir un tel bloc de mots similaires d'une ou plusieurs chaînes dont nous verrons par la suite qu'elle conduit à des algorithmes exacts bien plus efficaces que ceux utilisant la définition 3.4.6 établie à partir d'une fonction de score. L'introduction d'une contrainte de quorum q (voir la définition 3.3.5) est en outre bien plus aisée à traiter. La définition obtenue est alors la suivante [Soldano *et al.*, 1995] :

Définition 3.4.10 *Étant donné une chaîne $s \in \Sigma^n$ (qui est la concaténation de N chaînes), une relation R sur Σ, un entier q entre 2 et N représentant la contrainte de quorum et un autre entier k, les groupes de mots similaires de longueur k dans s sont les cliques maximales C_k de R_k qui satisfont la contrainte q.*

R peut bien sûr être égal à l'identité, ou peut être une relation d'équivalence. Dans le premier cas, les groupes de mots similaires sont en fait des ensembles de mots identiques. Cette définition de similarité entre mots est celle utilisée dans les algorithmes de Karp, Miller et Rosenberg [Karp *et al.*, 1972], Crochemore [Crochemore, 1981] [Crochemore and Rytter, 1991], Landraud [Landraud *et al.*, 1989] et Martinez [Martinez, 1983]. La relation d'identité a été étendue à une relation de similarité R quelconque, c'est-à-dire non nécessairement transitive, par Brutlag [Brutlag *et al.*, 1990] et Cobbs [Cobbs, 1994] pour le cas du deux à deux et par Soldano [Soldano *et al.*, 1995] pour le cas du multiple. Lorsque les chaînes représentent un codage linéaire de structures protéiques, et R une relation entre paires d'angles codées sur les nœuds d'une grille comme indiqué dans la section 3.2.3.2.2, alors les cliques maximales de R_k représentent tous les motifs structuraux contigus de longueur k d'un ensemble de protéines [Pothier, 1993] [Sagot *et al.*, 1995b].

Cette approche comporte une inflexibilité d'une autre sorte que celle vue précédemment puisqu'elle ne permet dans sa définition de similarité ni les substitutions non conservatives (i.e. celles impliquant des éléments non en relation par R) ni les trous. Avant de présenter notre travail qui a porté sur cet aspect-là justement, nous voulons discuter une dernière définition de similarité entre mots. Celle-ci ne résoud qu'en partie les problèmes soulevés par les deux premières mais elle représente une approche qu'il est essentiel de connaître car elle fournit de très bons résultats en pratique. C'est souvent contre ces résultats-là que nous avons mesuré les notres, nous reviendrons là-dessus dans le chapitre consacré aux illustrations. Cette approche fait fortement appel à divers critères de signification statistique, et c'est cela qui en fait à la fois son intérêt et résume toute la difficulté que nous pouvons avoir à bien saisir ce que la mesure de ressemblance utilisée capte vraiment.

3.4.3.3 Optimisation stochastique

Cette approche utilise pour mesurer la ressemblance d'un ensemble de mots une fonction d'optimisation stochastique qui n'est pas tout à fait immédiate à comprendre. Nous n'en fournissons ici qu'une idée très générale. La mesure de ressemblance que nous schématisons ci-dessous est donc présentée sous son aspect le plus simple, celui que donne Lawrence dans deux articles principaux [Lawrence *et al.*, 1993] [Lawrence and Reilly, 1990] et qui s'inspire de la formule utilisée par Hertz et Stormo [Cardon and Stormo, 1992] [Hertz *et al.*, 1990].

Définition 3.4.11 *Étant donné un ensemble de chaînes s1,s2,...,sN, le groupe de mots u1,u2, ...,uN dans s1,s2,...,sN respectivement ayant même longueur k et que l'on considère comme étant le plus similaire entre eux est celui qui maximize la valeur de la formule :*

$$\log L = \sum_{i=1}^{k} \sum_{j \in \Sigma} n_{ij} \log \frac{q_{ij}}{p_j}$$

où Σ est l'alphabet des monomères (nucléotides ou acides aminés), n_{ij} est le nombre de fois où le monomère j apparaît dans la colonne i de l'alignement sans trous des mots, q_{ij} est la fréquence d'apparition du monomère j à cette position et p_j est la fréquence d'apparition du monomère j sur les chaînes ailleurs que dans les mots du groupe.

Intuitivement, cette définition a pour base la quantité d'information que peut représenter un groupe de mots similaires. Cette information est résumée en une matrice qu'on espère spécifique du signal [Cardon and Stormo, 1992]. La ressemblance entre mots présents dans des chaînes est ainsi fonction de l'intensité du signal biologique qu'ils portent.

Observons toutefois que la mesure de la ressemblance que cette définition donne s'appuie sur des fréquences d'apparition observées. En particulier, il faut estimer la valeur de q_{ij} qui devrait indiquer la fréquence d'apparition du monomère j à la position i de tous les mots représentant un certain signal biologique, d'après la seule valeur connue qui est celle des n_{ij} indiquant le nombre de fois où le monomère j apparaît à la position i dans les exemples dont on dispose pour caractériser ce signal. Nous retrouvons donc ici la même difficulté rencontrée lorsque nous avons discuté du mode de construction des matrices de substitution pour les acides aminés (voir section 3.2.2.2) et de l'attribution d'une pondération aux trous dans un alignement (voir section 3.2.4.1.2). Elle devient plus critique dans ce cas puisqu'en général le nombre d'exemples (de chaînes) desquels nous devons extraire une information peut être relativement petit. Si par exemple q_{ij} est choisi proportionnel à n_{ij} (ce qui correspond à la façon la plus naturelle de procéder), nous pourrions avoir $q_{ij} = 0$ si l'acide aminé j n'est jamais observé dans l'ensemble des exemples dont nous disposons. La manière la plus classique de contourner cette difficulté consiste à employer des 'régularisateurs' qui peuvent être vus comme des rectificateurs des valeurs observées pour n_{ij} sur les exemples. Ces régularisateurs produisent ainsi des estimateurs pour q_{ij} (et d'une manière similaire non montrée ici pour p_j) appelées valeurs postérieures (en anglais 'posterior counts') et notées x_{ij}. Il existe plusieurs types de régularisateurs. Les trois principaux sont les pseudo-comptages (en anglais 'pseudo-counts'), les matrices de substitutions et les mélanges de Dirichlet. Nous ne parlerons que des deux premières, une référence à la troisième peut être trouvée dans l'article de Karplus [Karplus, 1995] dont nous avons adopté également les définitions concernant les pseudo-comptages et les matrices de substitution.

Les pseudo-comptages sont des régularisateurs non nuls qui dépendent du monomère en question et qui sont additionnés aux n_{ij}. On obtient ainsi :

$$x_{ij} = n_{ij} + z_j.$$

Le plus souvent, z_{ij} est choisi égal à $a.f_j$ où f_j est la fréquence d'apparition du monomère j partout sur les chaînes (donc $f_j \neq p_j$) et a une constante. C'est le régularisateur utilisé par Lawrence [Lawrence *et al.*, 1993]. Ce type de pseudo-comptage est appelé pseudo-comptage de bruit de fonds (en anglais 'background pseudo-counts'). Il fait référence aussi à la méthode de prédiction de Bayes.

Les régularisateurs qui emploient une matrice de substitution $\mathcal{M}$, et qui ne sont donc applicables qu'aux protéines, produisent des valeurs postérieures qui sont une combinaison linéaire des valeurs observées pour n_{ij} :

$$x_{ij} = \sum_k \mathcal{M}(k,j).n_{ik}.$$

Enfin les deux approches sont le plus souvent combinées pour produire des valeurs postérieures définies par :

$$x_{ij} = \sum_k \mathcal{M}(k,j).n_{ik} + z_j.$$

Karplus montre que les matrices de substitution seules sont les meilleurs régularisateurs pour estimer les valeurs de n_{ij} en présence d'un seul exemple mais qu'elles sont plus mauvaises que les pseudo-comptes au-delà de trois exemples. Il montre également que les mélanges de Dirichlet sont les meilleurs régularisateurs de tous mais ce sont les plus coûteux à calculer.

Il faut savoir également que des mesures plus complexes que celle donnée dans la définition sont possibles lorsque l'on veut identifier plusieurs blocs de similarité, modifier la longueur des mots similaires, permettre la présence de trous ou introduire une contrainte de quorum [Cardon and Stormo, 1992].

Toutes ces mesures font appel à des modèles que l'on peut voir comme des objets externes dont on se sert pour comparer les objets concrets que sont les mots. Dans le cas de Lawrence, les modèles sont des matrices de fréquences d'apparition des monomères aux positions dans les chaînes où se situe un signal. Dans les méthodes utilisant des HMM ('Hidden Markov Model') [Krogh *et al.*, 1994], les modèles sont des automates stochastiques. Nous pouvons donc considérer que les approches regroupées dans cette section se trouvent entre les mesures directes de comparaison et les indirectes. Les objets externes avec lesquels nous travaillons ne sont cependant pas les mêmes et, surtout, la manière de les obtenir est très différente. Les notres sont en effet construits par récurrence tandis que la taille des matrices de Lawrence et celle des automates stochastiques est le plus souvent fixée à l'avance, ou du moins estimée. En conséquence, les algorithmes utilisant la définition de similarité donnée ici, ou les formules que l'on peut trouver dans les articles sur les HMM, emploient tous des heuristiques pour établir ces matrices ou automates. Il est important cependant de noter que la raison n'est pas seulement qu'ils ne seraient pas efficaces en pratique s'il en était autrement, mais est propre à la philosophie même de cette approche. Nous reviendrons là-dessus dans le chapitre suivant. Avant d'examiner les objets externes que nous utilisons et qui conduisent à des algorithmes exacts de recherche de blocs de similarité, nous voulons cependant conclure cette section en parlant des diverses représentations qu'il est possible de donner des groupes de mots similaires obtenus par comparaison directe de ces mots entre eux.

3.4.4 Représentation des blocs obtenus par comparaison directe des mots

En effet, l'exploration de la ressemblance d'un ensemble d'objets dont nous pensons qu'ils exercent une même activité a en fin de compte pour but de caractériser ce que les objets ont en commun. L'idée est que ces caractérisations, si elles sont suffisamment précises et complètes, devraient nous permettre alors d'identifier d'autres macromolécules que celles de l'ensemble initial possédant la même activité. Une fois les traits communs repérés, dans le cadre de ce travail il s'agit de mots, il faut donc encore trouver un moyen de représenter l'information qu'ils contiennent afin de pouvoir s'en servir pour localiser ces traits ailleurs. Ainsi que nous l'avons mentionné à la fin du chapitre 2, l'idéal serait que le trait commun avec sa représentation soit non seulement caractéristique des molécules jouant un même rôle, mais qu'il leur soit aussi spécifique. Repérer un trait spécifique est bien sûr beaucoup plus difficile à réaliser, et il est quasiment impossible de vérifier que nous y sommes bien parvenus.

Les représentations qui ont été faites de ces groupes de mots similaires dans le cas où ceux-ci ont été comparés de façon directe sont aussi diverses que les définitions et les moyens élaborés pour les obtenir. La terminologie employée est aussi très confuse et ambiguë, certains termes ayant souvent été employés avec des significations différentes.

Nous pouvons cependant classer ces représentations en cinq grandes catégories que sont les alignements, les consensus, les expressions régulières de portée variable, les tableaux de propriétés et de relations et enfin les automates stochastiques auxquels nous avons fait référence dans la section précédente.

Les alignements sont la manière la plus simple et immédiate de représenter ces mots communs. C'est aussi à partir des alignements que la plupart des autres représentations sont établies. Les consensus et les expressions régulières en représentent ainsi des formes plus com-

pactes qui résument et parfois simplifient considérablement l'information contenue dans un alignement, et facilitent la recherche de cette information ailleurs.

La catégorie des consensus est elle-même divisée en deux sous-catégories, celle des matrices ou des profils, et celle des motifs consensuels. Les matrices de consensus [Goldstein and Waterman, 1994] [Schneider *et al.*, 1986] [Stormo, 1990] [Tatusov and Koonin, 1994] [Tatusov *et al.*, 1994] ne sont rien d'autre en fait que des tableaux où chaque entrée présente, pour chaque symbole de l'alphabet, la probabilité de trouver ledit symbole à une position donnée dans un groupe de mots alignés. Les profils [Gribskov *et al.*, 1987] [Gribskov *et al.*, 1988] [Gribskov *et al.*, 1990] contiennent en outre la probabilité d'observer un trou à une position quelconque dans l'alignement. Les motifs consensuels quant à eux sont des mots sur le même alphabet Σ que celui servant à représenter les macromolécules. Ils ont reçu des noms divers dans la littérature : motif [Bairoch, 1992] [Barton and Sternberg, 1990] [Posfai *et al.*, 1989] [Smith *et al.*, 1990] [Smith and Smith, 1990], motif consensuel [Galas *et al.*, 1985] [Hertz *et al.*, 1990] [Waterman, 1984b] [Waterman *et al.*, 1984] [Waterman, 1986] [Waterman, 1989] [Waterman, 1990], signature [Sheridan and Venkataraghavan, 1992]. Nous pouvons les classer en fait aussi parmi les expressions régulières dont ils représentent la forme la plus simple. Les formes plus complexes englobent les motifs dégénérés définis sur une partition de $\mathcal{P}(\Sigma) \setminus \{\Sigma\}$ [Galas *et al.*, 1985] [Smith and Smith, 1990] [Waterman, 1984b] [Waterman *et al.*, 1984] [Waterman, 1989], ceux définis sur un sous-ensemble quelconque de $\mathcal{P}(\Sigma) \setminus \{\Sigma\}$ [Taylor, 1986c] [Taylor, 1986b] [Taylor, 1989] ou de $\mathcal{P}(\Sigma)$ [Neuwald and Green, 1994] (dans ce second cas on dit que le joker $\{\Sigma\}$ — en anglais 'wild card' ou 'don't care symbol' — est autorisé) et enfin les expressions régulières complètes. Des exemples de consensus (matrice et motif) et d'expressions régulières sont données dans la figure 3.24.

Les représentations peuvent être plus sophistiquées et, dans le cas des protéines, essayer d'inclure une information structurale indiquant les propriétés physico-chimiques des mots et les relations proches ou distantes reliant chaque position du groupe aux autres. Des illustrations de ce type de représentation peuvent être trouvées dans l'article de Sali [Sali and Blundell, 1990] et dans celui d'Overington [Overington *et al.*, 1990]. Ces représentations ne sont possibles que si pour au moins une des chaînes de l'ensemble que l'on a comparé la structure tertiaire de la protéine est connue par ailleurs (par cristallographie par exemple). Enfin, les groupes de mots similaires identifiés par optimisation stochastique sont représentés dans leur forme la plus simple par des matrices [Lawrence *et al.*, 1993] [Lawrence and Reilly, 1990] ou dans leur forme plus complexe par un automate non déterministe dont les transitions sont données par des probabilités [Krogh *et al.*, 1994]. Les automates comme les matrices sont établis non pas à la fin de la recherche comme cela est le cas de toutes les autres représentations, mais au cours de celle-ci et sont ainsi modifiés en permanence. Ils servent donc à la fois à résumer des ensembles de mots et à construire ces ensembles. C'est dans ce sens également que nous avons dit que cette approche par optimisation stochastique est à mi-chemin entre les mesures directes de comparaison et les indirectes que nous avons élaborées et dont nous allons parler maintenant.

3.4.5 Résumé des mesures directes et locales de la ressemblance

Un résumé des mesures directes et locales de la ressemblance est donné dans le tableau ci-dessous.

Soit $\Sigma = \{a, b, c\}$. Nous présentons ci-dessous quelques-unes des manières les plus couramment utilisées pour représenter d'une manière condensée un groupe de mots similaires :

- matrice consensus :

position	1	2	3	4	5
a	0.70	0.10	0.50	0.20	0.85
b	0.20	0.10	0.40	0.00	0.05
c	0.10	0.80	0.10	0.80	0.10

- motif simple : $abaca$
- motif dégénéré, joker $\{\Sigma\}$ non autorisé :
  - les ensembles ne peuvent être que des classes: $[ab]b[ab]ca$
  - les ensembles peuvent être quelconques : $[ab][bc][c][a]$
- motif dégénéré, joker autorisé : $[ab]n[b]nn[c]$ où n est le joker $[abc]$
- expression régulière : $[ab]n\star[ab]$ où $n = [abc]$ et $\star$ représente un nombre quelconque de symboles.

Figure 3.24: Exemples de représentations de groupes de mots similaires.

Mesures Directes et Locales de la Ressemblance	
Caractéristiques principales	Référence aux définitions
optimisation d'une fonction de score	définition 3.4.6
recherches combinatoires exactes utilisation d'une relation R de similarité	définition 3.4.10
optimisation stochastique	définition 3.4.11

Tout ce que nous avons présenté jusqu'ici a été des définitions de similarité (dissimilarité) proposées par d'autres que nous. Les définitions que nous établissons à partir de maintenant concernent toutes celles que nous avons élaborées dans le cadre de ce travail.

3.5 Mesures indirectes de la ressemblance

3.5.1 Introduction aux modèles : à la fois des constructeurs et des résumés de la ressemblance

Les définitions de similarité entre mots que nous introduisons maintenant ont pour caractéristique principale de faire appel à des objets externes contre lesquels les mots dans les chaînes sont comparés. Ces objets externes sont à la fois la représentation résumée des blocs de similarité et leurs constructeurs. Ils sont donc obtenus en même temps que les blocs sont recherchés et toute définition de similarité entre mots est ainsi une définition de similarité entre un ensemble de mots et un certain objet externe (voir la figure 3.17).

Ces définitions essayent toutes de répondre, de diverses façons possibles, à certains problèmes que peuvent présenter les trois définitions précédentes. Elles permettent donc des erreurs (substitutions non conservatives et/ou trous) et l'utilisation d'une contrainte de quorum tout en

conduisant à des algorithmes qui sont exhaustifs, exacts et relativement efficaces. Tous les blocs de mots similaires suivant une de ces définitions et qui satisfont une contrainte de quorum q sont ainsi localisées en un temps qui demeure dans la plupart des cas raisonnable.

Les raisons pour lesquelles plusieurs définitions ont été établies et non pas une seule sont à la fois historiques et pragmatiques. La raison historique est celle à laquelle nous avons déjà fait allusion dans l'introduction. Tout au long de ce travail, nous avons essayé de modéliser la notion de ressemblance entre macromolécules biologiques. Au fur et à mesure que nous avancions, et que des premiers résultats étaient obtenus, nos idées ont changé, et des nécessités nouvelles sont apparues. Ces idées évoluent encore et il est vraisemblable qu'elles conduisent à de nouvelles définitions.

La raison pragmatique est liée au fait qu'il n'est sans doute pas réaliste d'espérer qu'une seule définition permette de traiter tous les cas auxquels nous pouvons être confrontés. Chaque définition peut ainsi être plus adaptée à la reconnaissance de traits communs particuliers. Pour l'instant, nous allons nous contenter de donner ces définitions, en essayant toujours cependant à chaque fois de les motiver biologiquement.

Ces définitions peuvent être divisées en deux grandes catégories : celles qui comptent les erreurs et celles qui affectent un score à ceux-ci. Nous disons des premières qu'elles travaillent avec une notion de distance basée sur des symboles, et de la dernière qu'elle utilise une notion de distance basée sur des mots.

Les objets externes employés sont également de deux types : il s'agit soit de mots définis sur le même alphabet que les chaînes que l'on compare, soit des produits cartésiens d'ensembles d'une couverture établie à partir de cet alphabet. Dans tous les cas, ces objets sont appelés des modèles. La représentation qu'ils fournissent d'un ensemble de mots similaires correspond ainsi soit aux motifs simples soit aux motifs dégénérés des mesures directes. Il est sans doute possible d'imaginer d'autres modèles.

Observons finalement que, dans tout le reste de ce chapitre, nous allons adopter la notation de la section 3.4.3.2. Chaque mot présent dans une des chaînes d'un ensemble $s1$,$s2$,...,sN va ainsi être identifié par sa position (son indice) dans la chaîne s qui est la concaténation de $s1$,$s2$,...,sN. Par ailleurs, la contrainte de quorum qui va jouer sur un modèle est bien sûr la contrainte sur les blocs de la définition 3.3.5 puisqu'un modèle identifie toujours un bloc de mots similaires.

3.5.2 Les différentes définitions de similarité travaillant avec des modèles

3.5.2.1 Définitions utilisant une notion de distance basée sur des symboles

3.5.2.1.1 Des modèles qui sont des mots — une application à l'ADN et à l'ARN

Nous avons indiqué au début de ce chapitre (section 3.2.2.1) que lorsqu'il s'agit d'analyser des macromolécules d'ADN ou d'ARN, il est traditionnel de considérer que deux nucléotides sont soit identiques, soit différents sans moyen terme entre les deux. Des relations d'équivalences ou de similarités sont rarement établies dans ce cas. De telles relations sont pourtant possibles et envisageables dans certaines situations, mais ici nous décidons que nous allons simplement autoriser un certain nombre d'erreurs entre les mots que nous définissons comme étant similaires. En fait, comme nous venons de le voir, les erreurs entre mots vont être introduites de manière indirecte : c'est ainsi entre chaque mot d'un ensemble et un même modèle qu'un certain nombre d'erreurs va être permis. Comme aucune relation de similarité n'est établie entre les nucléotides, la définition d'un modèle que nous employons dans ce cas est celle d'un mot sur le même alphabet Σ que celui des monomères [Sagot *et al.*, 1995a] :

Soient $\Sigma = \{A, B, C\}$ et $e = 1$. Étant donné le modèle :

$$m = ABCABC$$

le mot $u = ACABC$ est une L-image de m avec une délétion (premier B) et le mot $v = ABCABBC$ est une L-image de m avec une insertion (second B). Les deux occurrences sont notées (1, 0, 1, 0) et (1, 0, 0, 1) respectivement.

Figure 3.25: Convention pour noter le nombre de délétions et insertions.

Définition 3.5.1 *Un modèle m de longueur k est un élément de Σ^k.*

La relation entre un modèle et des mots présents dans s est alors donnée par :

Définition 3.5.2 *Étant donné un modèle m, m est dit être L-présent dans une chaîne s à la position i s'il existe au moins un mot u qui se trouve situé à cette position dans s et qui est tel que $distance_L(u,m) \leq e$ où $distance_L$ est la distance de Levenshtein entre u et m. Nous appelons u une L-image de m.*

Soit $d_s + d_d + d_i = distance_L(u,m)$ où d_s, d_d et d_i sont le nombre de substitutions, délétions et insertions respectivement (par convention, nous notons d_d et d_i respectivement le nombre de délétions et insertions réalisées sur u — voir la figure 3.25).

Nous disons que (i,d_s,d_d,d_i) est une L-occurrence de m dans s. Nous avons $\mid u \mid = \mid m \mid - d_d + d_i$.

Finalement, nous notons $L(m)$ l'ensemble de toutes les L-occurrences de m dans s, c'est-à-dire :

$$L(m) = \{(i,d_s,d_d,d_i) \mid d = d_s+d_d+d_i \text{ est une } distance_L(u,m) \text{ et } d \leq e\}.$$

Exemple 3.5.1 *Voir la figure 3.26.*

Observons que la décomposition en d_s, d_d et d_i peut ne pas être unique. Un modèle peut donc être L-présent dans s à la position i plus d'une fois, avec différentes L-images. Il peut même être présent pour la même image u avec différentes L-occurrences.

Par ailleurs, la relation qui est établie entre un modèle et son image induit une relation entre les images elles-mêmes, c'est-à-dire entre les mots de s, puisque la distance de Levenshtein entre deux quelconques d'entre eux est au plus $2e$.

Il convient d'observer toutefois que la réciproque n'est pas vraie, à savoir, étant donné un ensemble de mots tels que la distance entre deux quelconques d'entre eux est au plus $2e$, il n'existe pas toujours un modèle avec lequel tous ces mots soient en relation. Un exemple de ce fait est donné ci-dessous :

Exemple 3.5.2 *Soient $\Sigma = \{A, C, G, T\}$ et $e = 1$. Soient les mots $AGCG$, $ACCC$ et $ATCT$. La distance de Levenshtein entre deux quelconques de ceux-ci est toujours égale à $2e = 2$, mais il est clair qu'il n'existe aucun modèle m tel que la distance de chacun des mots à m soit au plus 1.*

Soient $\Sigma = \{A, C, G, T\}$ et $e = 1$. Soient les trois chaînes :

$s1 = CAAATTCATA\underline{ACAAATGGAGGA}GGATGTATATGTATCCGGA$
$s2 = TATTTAACCGTGACA\underline{ACAAGGAGGA}AACGTAATGATGAATC$
$s3 = TGGCATAAGCTGAT\underline{ACATAGGAGGA}CGAATATGACTTGGAG$

Le plus long modèle trouvé dans les trois chaînes est le modèle :
$ACAAAGGAGGA$
situé à la position 11 de $s1$ (avec 1 insertion), 15 de $s2$ (avec 1 délétion) et 15 de $s3$ (avec 1 substitution).

Figure 3.26: Exemple d'un modèle et de ses L-occurrences.

Des modèles qui sont des mots et la distance de Levenshtein nous fournissent une façon de définir une relation de similarité entre des mots présents dans des chaînes.

Définition 3.5.3 *Étant donné une chaîne $s \in \Sigma^n$ (qui est la concaténation de N chaînes), un entier q entre 2 et N représentant une contrainte de quorum et un autre entier k, les groupes de mots similaires dans s sont les ensembles de L-occurrences de tous les modèles m de longueur k satisfaisant cette contrainte.*

Notons que, bien qu'établie de façon différente, cette définition est très similaire à celle utilisée par Waterman et introduite avant nous dans la littérature [Galas *et al.*, 1985] [Waterman, 1984b] [Waterman *et al.*, 1984] [Waterman, 1986] [Waterman, 1989] [Waterman, 1990]. Sa façon d'obtenir les modèles, et donc les ensembles de mots similaires, est toutefois très différente et d'une application pratique souvent plus limitée, ainsi que nous le montrons dans le chapitre 5.

3.5.2.1.2 Des modèles comme produits d'ensembles d'une couverture — une application aux protéines

Dans le cas des protéines, nous avons vu qu'il existe des relations de similarité entre les acides aminés (section 3.2.2.2) et que la complexité de telles relations est captée au mieux à l'aide d'une couverture. La définition d'un modèle que nous employons dans ce cas utilise donc le concept d'une couverture C de l'alphabet et nous avons [Sagot *et al.*, 1995d] :

Définition 3.5.4 *Un modèle M de longueur k est un élément de C^k. Nous notons $\mid M \mid$ la longueur de M.*

Avant d'indiquer quelle est la relation existant entre un modèle et des mots dans s, nous introduisons deux notations qui vont nous être utiles :

Notation 3.5.1 *Étant donné $u = u_1...u_k \in \Sigma^k$ et un modèle $M = M_1...M_k \in S^k$, nous disons que u est une <u>instance</u> de M si, pour tout $i \in \{1,...,k\}$, $u_i \in M_i$.*
En d'autres termes, u est une instance de M si $u \in M$.

Soient $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ et C la couverture de Σ donnée par :

$S_1 = \{I, L, V, M\}$
$S_2 = \{A, G, S, C, T\}$
$S_3 = \{K, R, H\}$
$S_4 = \{K, R\}$
$S_5 = \{Y, W, F, H\}$
$S_6 = \{Y, W, F\}$
$S_7 = \{D, E\}$
$S_8 = \{Q, N\}$
$S_9 = \{P\}$
$S_{10} = \{C\}$
$S_{11} = \{W\}$

Soient les quatre chaînes suivantes :

$s1 = VMGPIKDM\underline{VHITHGPIG}CSFYTWGGRRFKSKPENGTGLNFNEYV$
$s2 = CAYAGSKGVVWGPIKDM\underline{IHISHGPVG}CGQYSRAGRRNYYIGTTGV$
$s3 = EFCGGHTHAISRYGLEDMLPAN\underline{VRMIHGPGC}PVCVLPAGRIDMAI$
$s4 = MKLAHWMYAGPAHIGTLRVASSFKN\underline{VHAIMHAPLG}DDYFNVMR$

Les plus longs modèles trouvés dans ces chaînes sont les modèles suivants :

$S_1S_3S_1S_1S_5S_2S_9S_1S_2$

situé à la position 9 de $s1$ (avec 1 substitution), 18 de $s2$ (avec 1 substitution), 23 de $s3$ (deux fois, avec 1 substitution et 1 délétion) et 26 de $s4$ (avec 1 insertion), et :

$S_1S_3S_1S_1S_3S_2S_9S_1S_2$

situé à la position 9 de $s1$ (avec 1 substitution), 18 de $s2$ (avec 1 substitution), 23 de $s3$ (deux fois, avec 1 substitution et 1 délétion) et 26 de $s4$ (avec 1 insertion).

Figure 3.27: Exemple de modèles et de leurs EL-occurrences

Notation 3.5.2 *Étant donné $u \in \Sigma^*$ et un modèle $M \in S^k$, nous appelons $\underline{distance_{EL}}$ entre u et M, notée $distance_{EL}(u, M)$, le nombre minimum de substitutions, délétions et insertions nécessaires pour passer de u à une instance de M ($distance_{EL}$ signifie 'distance Ensembliste de Levenshtein').*

La relation entre un modèle et un mot est alors définie par :

Définition 3.5.5 *Étant donné un entier e et un modèle M, nous disons que M est $\underline{EL\text{-présent}}$ dans s à la position i s'il existe au moins un mot u situé à cette position dans s tel que $distance_{EL}(u, M) \leq e$. Nous appelons u une $\underline{EL\text{-image}}$ de M.*

Soit $d_s + d_d + d_i = distance_{EL}(u, M)$ où d_s, d_d et d_i sont le nombre de substitutions, délétions et insertions respectivement entre u et l'instance la plus proche de M (à nouveau ici, par convention, nous notons d_d et d_i respectivement le nombre de délétions et insertions réalisées sur u). Observons que cette décomposition peut ne pas être unique. Nous disons que (i,d_s,d_d,d_i) est une $\underline{EL\text{-occurrence}}$ de M dans s et nous avons $\mid u \mid = \mid M \mid - d_d + d_i$.

Soient Σ l'alphabet des acides aminés et C la couverture donnée dans la figure 3.27. Supposons par ailleurs qu'aucune erreur ne soit autorisée entre un modèle et ses occurrences et que la contrainte de quorum q soit égale à 2.
Un exemple de situation où :

- $M_1 \subseteq M_2$ et
  - $EL(M_1)$ égal à $EL(M_2)$ est donnée par les chaînes :
    $s1 = CKD$
    $s2 = GRE$
    $s3 = SRD$
    et les modèles valides $S_2S_4S_7$ et $S_2S_3S_7$;
  - $EL(M_1)$ strictement inclus dans $EL(M_2)$ est donnée par les chaînes :
    $s1 = CKD$
    $s2 = GRE$
    $s3 = SHD$
    et les modèles valides $S_2S_4S_7$ et $S_2S_3S_7$;
- $M_1 \cap M_2 \neq \emptyset$ avec $M_1 \neq M_2$ et
  - $EL(M_1)$ égal à $EL(M_2)$ est donnée par les chaînes :
    $s1 = CHD$
    $s2 = GHE$
    $s3 = SHD$
    et les modèles valides $S_2S_3S_7$ et $S_2S_5S_7$;
  - $EL(M_1)$ strictement inclus dans $EL(M_2)$ est donnée par les chaînes :
    $s1 = CHD$
    $s2 = GHE$
    $s3 = SYD$
    et les modèles valides $S_2S_3S_7$ et $S_2S_5S_7$.

Figure 3.28: Exemple des rapports pouvant exister entre les ensembles d'occurrences des modèles présents dans un ensemble de chaînes.

Finalement, nous notons $EL(M)$ l'ensemble des EL-occurrences de M dans s. Nous avons alors :

$$EL(M) = \{(i,d_s,d_d,d_i) \mid d = d_s+d_d+d_i \text{ est une distance}_{EL} \text{ et } d \leq e\}.$$

Exemple 3.5.3 *Voir la figure 3.27.*

À nouveau ici, un modèle peut être EL-présent dans s à la position i plus d'une fois avec différentes EL-images (voir la figure 3.27), ou, pour la même image u, avec différentes EL-occurrences.

De même, la relation qui est établie entre un modèle et ses images induit une relation entre les images elles-mêmes, c'est-à-dire entre les mots de s, puisque la distance de Levenshtein, 'simple' cette fois, entre deux quelconques d'entre eux est au plus $2e$. La réciproque n'est pas vraie ainsi que nous l'avons montré dans la section précédente. L'exemple 3.5.2 donné alors s'applique également ici, il suffit de prendre $C = CI = \{\{a\} \mid a \in \Sigma\}$.

Quant à la couverture que nous utilisons, elle est *a priori* quelconque. En particulier, une couverture peut contenir un sous-ensemble qui est totalement inclus dans un autre comme cela est le cas des exemples donnés dans les figures 3.12 et 3.27. Ce qui n'est par contre absolument pas permis par la définition de similarité entre mots donnée dans cette section sont les couvertures qui contiennent comme un de leurs sous-ensembles tout l'alphabet — ce que nous avons appelé le joker (rappelons qu'en anglais il s'agit du 'wild card' ou du 'don't care symbol'). Il est aisé de voir pourquoi autoriser le joker représente un problème algorithmique complètement différent exigeant qu'une contrainte additionnelle soit imposée sur le nombre de fois où le joker est permis dans un modèle. Supposons en effet que le joker $\{\Sigma\}$ soit un des sous-ensembles de la couverture. Alors, étant donné un ensemble de chaînes et en faisant l'hypothèse qu'aucune erreur n'est autorisée, le plus long modèle qui est présent dans ces chaînes est trivialement le modèle $\{\Sigma\}^p$ où p est la longueur de la plus courte chaîne de l'ensemble. Des couvertures où le joker est permis mais avec une contrainte, et des couvertures plus combinatoires que celles que nous employons ici, y compris la couverture $\mathcal{P}(\Sigma)$, sont introduites dans la définition de la section suivante. Elles correspondent aux couvertures combinatoires pondérées mentionnées dans la section 3.2.3.3.1 et s'appliquent aussi bien à l'analyse des protéines qu'à celle des acides nucléiques.

Observons qu'il peut apparaître une certaine redondance de l'information captée par des modèles définis sur une couverture de l'alphabet. En effet, étant donné deux modèles M_1 et M_2, nous pouvons avoir les quatre situations suivantes :

- $M_1 \subseteq M_2$ et
  - $EL(M_1)$ égal à $EL(M_2)$;
    ou
  - $EL(M_1)$ strictement inclus dans $EL(M_2)$;
- $M_1 \cap M_2 \neq \emptyset$ avec $M_1 \neq M_2$ et
  - $EL(M_1)$ égal à $EL(M_2)$;
    ou
  - $EL(M_1)$ strictement inclus dans $EL(M_2)$.

Un exemple de chacun de ces cas est donné dans la figure 3.28. Nous pouvons ainsi définir deux types de non-maximalité de l'ensemble d'occurrences d'un modèle.

Définition 3.5.6 *Soient s1,...,sN des chaînes définies sur Σ^*, C une couverture de Σ et $M \in C^k$ un modèle de longueur k. Nous disons que :*

- *$EL(M)$ est intrinsèquement non-maximal s'il existe au moins un modèle $M' \in C^k$ tel que $M \subset M'$ et $EL(M) \subset EL(M')$;*
- *$EL(M)$ est non-maximal par rapport aux chaînes s1,...,sN s'il existe au moins un modèle $M' \in C^k$ tel que $M \cap M' \neq \emptyset$ avec $M \neq M'$ et $EL(M) \subset EL(M')$.*

Bien sûr, le premier point ne peut être vérifié que s'il existe S_i, $S_j \in C$ tel que $S_i \subset S_j$. Dans ce cas seulement, on peut définir un concept de non-minimalité d'un modèle par rapport à un autre.

Définition 3.5.7 *Soient M et M' deux modèles définis sur une couverture C de l'alphabet Σ. Si $M \subset M'$ et $EL(M) = EL(M')$, nous disons que le modèle M' est non-minimal par rapport au modèle M.*

Une telle redondance en termes d'ensembles d'occurrences n'est pas forcément mauvaise, au contraire, puisque nous sommes le plus souvent intéressés non seulement par les blocs de mots similaires que nous réussissons à identifier, mais aussi par leur représentation - les modèles. Exiger soit la maximalité d'un ensemble d'occurrences (intrinsèque ou liée aux chaînes que l'on compare), soit la minimalité d'un modèle peut être important cependant. Cela introduit toutefois une contrainte supplémentaire au niveau à la fois de la définition de similarité entre mots et de l'algorithme qui permet de localiser ces mots. Cette contrainte étant intimement liée à l'algorithme lui-même, elle est présentée dans le cas de la minimalité des modèles au chapitre 5. Nous verrons alors que la minimalité que nous obtenons en pratique est en fait plus restreinte que celle de la définition 3.5.7. Nous l'appelons 'minimalité gauche-droite'. Quant à la contrainte de maximalité sur les ensembles d'occurrences, nous en discutons plus loin lorsque nous montrons le rapport entre les modèles et les cliques maximales d'une relation R_k telles que définies dans la section 3.4.3.2. Ce rapport ne peut être établi bien sûr que si aucun élément de la couverture C n'est entièrement inclus dans un autre, ce qui en pratique sera le plus souvent le cas.

Des modèles qui sont des produits d'ensembles d'une couverture et la distance Ensembliste de Levenshtein nous fournissent donc une autre définition d'un groupe de mots similaires.

Définition 3.5.8 *Étant donné une couverture C de Σ, une chaîne $s \in \Sigma^n$ (qui est la concaténation de N chaînes), un entier q entre 2 et N représentant une contrainte de quorum et un autre entier k, les groupes de mots similaires dans s sont les ensembles de EL-occurrences de tous les modèles M de longueur k qui satisfont la contrainte q.*

Observons finalement qu'un exemple particulier d'une couverture est celle composée uniquement de singletons. Il est clair qu'il n'existe pas vraiment de différence entre des modèles comme produits cartésiens d'ensembles d'une telle couverture et des modèles définis comme des mots comme dans la section précédente. En termes des méthodes utilisées pour trouver les groupes de mots similaires, les idées de base sont exactement les mêmes pour les deux définitions et seule la version la plus générale de l'algorithme pour des modèles comme produits cartésiens d'ensembles d'une couverture (non combinatoire et non pondérée) est présentée dans le chapitre 5.

3.5.2.1.3 Des modèles comme produits d'ensembles d'une couverture combinatoire pondérée — une application aux protéines et aux acides nucléiques

Nous allons introduire dans cette section une définition de ressemblance utilisant des modèles définis cette fois sur une couverture quelconque, qui peut aussi bien être $\mathcal{P}(\Sigma)$. La motivation est double. La première, et sans doute la plus importante, est de nature biologique. Elle représente le fait que même lorsqu'une activité macromoléculaire en général est identifiable à un ensemble de mots, et donc caractérisable par un modèle, toutes les positions dans cet ensemble de mots peuvent ne pas être impliquées dans l'activité en question. Cela a été maintes fois observé, entre autres par Posfai [Posfai *et al.*, 1989], Bashford [Bashford *et al.*, 1987], Rooman [Rooman *et al.*, 1990] [Rooman and Wodak, 1988], Smith [Smith *et al.*, 1990] et Neuwald [Neuwald and Green, 1994] dans le cas des protéines. Le modèle peut ainsi contenir des positions qu'on appelle non-spécifiques, c'est-à-dire des positions où la conservation d'un monomère, ou d'un type particulier de monomères, n'est pas essentielle. Autoriser la présence du joker dans un modèle permet de capter cette non-spécificité de certaines positions. La seconde motivation est liée à l'idée mentionnée dans la section 3.2.3.3.1 d'essayer de se libérer de tout point de vue dans la recherche de mots communs à un ensemble de chaînes. Cela signifie en particulier travailler avec plusieurs alphabets en même temps. Nous introduisons ici les outils nécessaires dans ce but. Le principal est ce que nous avons appelé une couverture combinatoire pondérée (voir la fin de la section 3.2.3.3.1). Les modèles que nous employons dans cette nouvelle définition de similarité entre mots sont donc des produits cartésiens d'ensembles d'une telle couverture [Sagot and Viari, 1996].

Définition 3.5.9 *Soit Σ l'alphabet des monomères. Une couverture combinatoire pondérée CCP de Σ est définie par :*

$$CCP = \{(S, w_S) \mid S \in \mathcal{P}^+(\Sigma) \text{ et } w_S \text{ est un entier non négatif}\} \text{ où } \mathcal{P}^+(\Sigma) = \mathcal{P}(\Sigma)\backslash\emptyset.$$

Définition 3.5.10 *Soit CCP une couverture combinatoire pondérée de Σ, un modèle $M = S_1...S_k$ est un élément de CCP^k si, pour tout $1 \leq i \leq k$, $S_i \in \mathcal{P}^+(\Sigma)$ et le nombre de fois que S_i apparaît dans M n'est pas supérieur à w_{S_i}.*

Exemple 3.5.4 *Soit $\Sigma = \{A, C, G, T\}$ et :*

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 1)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 1) \end{array} \right\}$$

Alors nous avons :
$\{A, C\}\{A, G\}\{A, T\}\{C, G\}\{C, T\}\{G, T\} \in CCP^6$
mais par exemple les modèles :
$\{A, C\}\{A, C\}\{A, T\}\{C, G\}\{C, T\}\{G, T\}$
et
$\{A, C\}\{A, G\}\{A, T\}\{C, T\}\{A, C, G, T\}\{A, C, G, T\}$
ne sont pas des éléments de CCP^6 (parce que les ensembles $\{A, C\}$ et $\{A, C, G, T\}$ respectivement sont présents deux fois dans le modèle).

Définition 3.5.11 *Étant donné une couverture combinatoire contrainte CCP, une chaîne $s \in \Sigma^*$ et un modèle $M \in CCP^k$, M est dit être ECL-présent à la position i si le mot u de*

Soient $\Sigma = \{A, C, G, T\}$ et :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 1)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 1) \end{array} \right\}.$$

Soient les trois chaînes :

$s1 = AAC\underline{TGTGAGCATGGTCA}TATTT$
$s2 = TAA\underline{TGTGAGTTAGCTCA}CTCAT$
$s3 = AAT\underline{TGTGACACAGTGCA}AATTC$

Un plus long modèle trouvé dans les trois chaînes est le modèle :

$\{T\}\{G\}\{T\}\{G\}\{A\}\{C, G\}\{A, C, T\}\{A, C, G, T\}\{A, T\}\{G\}\{C, G, T\}\{G, T\}\{C\}\{A\}$
situé à la position 4 de $s1$, $s2$ et $s3$.

Figure 3.29: Exemple d'un modèle et de ses ECL-occurrences.

longueur $|\ M\ |$ *situé à cette position dans* s *est tel que* $u \in M$. u *est appelé une ECL-image de* M *et nous disons que* (i, u) *est une ECL-occurrence de* M *dans* s.

Nous notons ECL(M) l'ensemble des occurrences de M *dans* s, *à savoir :*

$$ECL(M) = \{(i, u = s_i...s_{i+|M|-1})\ |\ u \in M\}$$

Observons que la notation ECL vient de 'Ensembliste Combinatoire de Levenshtein', le L étant justifié par le fait que, bien qu'il n'y ait pas de notion d'erreur (et donc de distance) dans cette définition, nous pouvons en introduire une, ce que nous ferons dans le chapitre 5 (section 5.4.2.3).

Exemple 3.5.5 *Voir la figure 3.29.*

Des modèles comme produits particuliers d'ensembles d'une couverture combinatoire pondérée nous donnent alors une nouvelle définition de la similarité entre mots.

Définition 3.5.12 *Étant donné une couverture combinatoire pondérée* CCP *de* Σ, *une chaîne* $s \in \Sigma^n$ *(qui est la concaténation de* N *chaînes), un entier* q *entre* 2 *et* N *représentant une contrainte de quorum et un autre entier* k, *les groupes de mots similaires dans* s *sont les ensembles de* ECL*-occurrences de tous les modèles* M *de longueur* k *qui satisfont la contrainte* q.

Observons que cette définition ressemble beaucoup à la précédente mais qu'algorithmiquement les façons de construire les modèles et obtenir ainsi les groupes de mots similaires sont différentes et sont donc présentées séparément dans le chapitre 5.

Il est très important de noter cependant également que les deux définitions peuvent être regroupées, en particulier que des erreurs peuvent être autorisées entre un modèle M défini

à partir d'une couverture combinatoire pondérée et ses ECL-occurrences. Pour des raisons de clarté à nouveau, nous avons maintenu les deux définitions séparées.

3.5.2.2 Définition utilisant une notion de similarité basée sur des sous-mots

Avec la dernière définition de similarité entre mots que nous présentons maintenant, nous revenons à des modèles plus simples puisque ceux-ci vont à nouveau être des mots sur le même alphabet que celui des chaînes que nous comparons [Sagot *et al.*, 1995c]. La définition d'un modèle utilisée dans cette section est donc identique à celle de la section 3.5.2.1.1.

La raison de ce retour en arrière est que nous allons travailler cette fois avec une notion de similarité entre mots (qui peut aussi être une distance) qui est elle-même établie sur des sous-mots d'une certaine longueur w - et non plus sur un simple comptage de symboles identiques (ou différents). La motivation a été donnée dans la section 3.2.4.1.1 où nous avons montré les deux philosophies différentes de la mesure de ressemblance. Les définitions antérieures suivent une de ces philosophies, celle d'une ressemblance basée sur un comptage d'erreurs ou d'identités, la définition de cette section s'appuie sur l'autre qui utilise un système de score et est en général plus flexible ainsi que le suggérait la figure 3.14.

Dans l'exemple de cette figure, les trous ne sont pas permis entre les mots. Nous commençons donc par présenter une définition où les trous ne sont pas pris en considération. Cela est fait dans le but de simplifier la présentation. La définition permettant des trous est plus complexe et est donnée à la fin de cette section.

Observons que l'idée de cette définition dans son cas le plus simple ressemble beaucoup à celle employée pour effectuer des comparaisons deux à deux par le plus populaire des programmes de recherche rapide sur banque, appelé BLAST [Altschul *et al.*, 1990]. Nous appelons de tels programmes des cribleurs (en anglais 'scans'). Ici nous l'appliquons au cas multiple dans le cadre d'un ensemble de chaînes. Nous présentons BLAST dans la section 4.2.2.1.2.1.

Définition 3.5.13 *Soient $u = u_1u_2...u_k \in \Sigma^k$, $m = m_1m_2...m_k$ un modèle de longueur k et $\mathcal{M}$ une matrice de similarité, nous notons :*

$$score_{\mathcal{M}}(u, m) = \sum_{i=1}^{k} \mathcal{M}(u_i, m_i).$$

Définition 3.5.14 *Étant donné une valeur numérique t, un modèle m et un entier w entre* 1 *et $\mid m \mid$, nous disons que m est w-présent dans s à la position i si le mot u de longueur $\mid m \mid$ situé à cette position dans s est tel que :*

$$score_{\mathcal{M}}(u_j...u_{j+w-1}, m_j...m_{j+w-1}) \geq t$$

pour tout $j \in \{1,...,\mid m \mid -w+1\}$, c'est-à-dire si tout sous-mot de longueur w de u possède un score supérieur ou égal à t avec le sous-mot correspondant de m.

Nous appelons u une w-image de m.

Nous disons que (i, u) est une w-occurrence de m dans s et nous notons $K(m)$ l'ensemble de toutes les w-occurrences de m dans s, à savoir,

$$K(m) = \{(i, u) \mid \forall j \in \{1, ..., \mid m \mid -w+1\}, score_{\mathcal{M}}(u_j...u_{j+w-1}, m_j...m_{j+w-1}) \geq t\}.$$

Finalement, nous notons $l_w(u, m) = score_{\mathcal{M}}(u_{|m|-w+1}...u_{|m|}, m_{|m|-w+1}...m_{|m|})$. Il s'agit simplement du score du dernier sous-mot de longueur w de u et m.

Soient Σ l'alphabet des acides aminés, $\mathcal{M}$ la matrice PAM250 de la figure 3.7, $w = 3$ et $t = 60$. Soient les trois chaînes :

$s1 = MTDKAPQKT\underline{HLNIVIICG}G$

$s2 = MGKEK\underline{HINLVVIWW}WGHVN$

$s3 = MKMPKDK\underline{HVNIVFICV}GEE$

alors le plus long modèle qui est w-présent dans les trois chaînes est le modèle $HINIVVICW$ de longueur 9 (à la position 10 de $s1$, 6 de $s2$ et 8 de $s3$).

Figure 3.30: Un exemple de modèle et de ses w-occurrences.

Exemple 3.5.6 *Voir la figure 3.30.*

Observons que lorsque la matrice utilisée est une matrice de distance (dans ce cas, l'inégalité dans la définition 3.5.14 est renversée), la relation qui est établie entre un modèle et ses w-images induit une relation entre les images elles-mêmes à cause de l'inégalité triangulaire.

Les considérations précédentes nous fournissent alors une dernière définition de la similarité entre un groupe de mots.

Définition 3.5.15 *Étant donné une chaîne $s \in \Sigma^n$ (qui est la concaténation de N chaînes), un entier q entre 2 et N représentant une contrainte de quorum, un autre entier k et une constante w entre 1 et k, les groupes de mots similaires de longueur k dans s sont les ensembles de w-occurrences de tous les modèles m de longueur k qui satisfont cette contrainte.*

Que se passe-t-il lorsque des trous sont permis? Rappelons (voir section 3.2.4.1.2) que nous travaillons avec des scores constants pour les trous.

Notation 3.5.3 *Soit $\mathcal{A}$ un alignement avec trous d'un modèle m avec un mot u. On appelle longueur d'une paire de sous-mots de $\mathcal{A}$ la longueur du sous-mot dans m.*

Exemple 3.5.7 *Soit l'alignement $\mathcal{A} = \begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} b \\ \lambda \end{pmatrix}\begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} \lambda \\ a \end{pmatrix}\begin{pmatrix} a \\ a \end{pmatrix}$ du modèle $m = abaca$ avec le mot $u = aacaa$. Les paires de sous-mots de longueur 3 de $\mathcal{A}$ sont les paires (aba, aa), (bac, ac) et $(aca, acaa)$.*

Définition 3.5.16 *Étant donné une valeur numérique t, un modèle m et trois entiers w, e_s et e_t avec $1 \leq w \leq \mid m \mid$, $0 \leq e_s < w$ et $e_s \leq e_t$, nous disons que m est w-présent dans s à la position i s'il existe un mot u situé à cette position dans s tel que l'alignement $\mathcal{A}$ de m et u ne comporte au total pas plus de e_t trous, tandis que celui de toute paire de sous-mots de longueur w de m de l'alignement $\mathcal{A}$ ne présente pas plus de e_s trous et a un score supérieur ou égal à t (où le score d'un alignement avec trou est celui donné dans la définition 3.2.20)*

Exemple 3.5.8 *En reprennant les données de la figure 3.30 avec de plus $e_s = 1$, $e_t = 2$ et w_λ (score d'un trou) $= 0$, le mot aacaa est une occurrence du modèle abaca car l'alignement $\mathcal{A} = \begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} b \\ \lambda \end{pmatrix}\begin{pmatrix} a \\ a \end{pmatrix}\begin{pmatrix} c \\ c \end{pmatrix}\begin{pmatrix} \lambda \\ a \end{pmatrix}\begin{pmatrix} a \\ a \end{pmatrix}$ possède deux trous en tout, celui de chaque sous-mot de longueur w de m présente au plus un trou et a un score égal à 6 pour la paire (aba, aa), à 6 pour la paire (bac, ac) et à 9 pour la paire $(aca, acaa)$.*

Exemple 3.5.9 *Un second exemple reprend celui de la figure 3.30 avec de plus* $e_s = 1$, $e_t = \infty$ *et* 0 *le score attribué à un trou. Le plus long modèle qui est w-présent dans les trois chaînes de la figure est alors le modèle* $KQHINIVVICWG$ *de longueur* 13 *(à la position* 6 *de* $s1$ *avec une insertion,* 3 *de* $s2$ *avec également une insertion et* 5 *de* $s3$ *sans trou).*

3.5.3 Résumé des mesures indirectes (et locales) de la ressemblance

Un résumé des mesures indirectes et locales de la ressemblance est donné dans le tableau ci-dessous.

Mesures Indirectes et Locales de la Ressemblance		
Types de modèles	Caractéristiques principales	Référence aux définitions
mots	distance de Levenshtein	définition 3.5.3
produits d'ensembles d'une couverture	distance ensembliste de Levenshtein	définition 3.5.8
produits d'ensembles d'une couverture combinatoire pondérée	appartenance au modèle	définition 3.5.12
mots	scores des sous-mots supérieurs à seuil	définition 3.5.15

3.5.4 Les mesures directes revues

Lorsque nous avons présenté les mesures directes dans la section 3.4, nous avons montré qu'une des approches possibles pour localiser des blocs de similarité dans un ensemble de chaînes consiste à travailler avec une relation R et avec les cliques maximales de cette relation. Aucun modèle n'est nécessaire et les mots sont en fait comparés directement entre eux. Implicitement cependant, les modèles sont présents.

En effet, si nous représentons un modèle comme une boîte avec une étiquette dans laquelle certains objets (des positions dans une chaîne s) sont déposés, nous pouvons voir les cliques maximales comme des boîtes noires qui n'ont pas besoin d'être étiquetées pour remplir leur fonction de regroupement des mots similaires. Et pourtant, si nous examinons la définition d'une clique maximale C_k, nous constatons qu'une telle clique possède par rapport à s un nom unique même si ce nom n'est pas utilisé. Ce nom est celui du k-produit des cliques de R de plus petite cardinalité qui le composent.

Exemple 3.5.10 *En reprennant les données de la figure 3.23, le nom de l'unique clique maximale de* R_2 *par rapport à* s *est égal à* $\{L, M\}\{I, L\}$.

Nous pouvons donc considérer que la définition 3.4.10 de similarité entre mots n'est qu'un cas particulier de la définition 3.5.8 puisque les cliques maximales d'une relation R (et les classes) ne sont elles-mêmes que des cas particuliers d'une couverture de Σ. Il n'y a qu'une réserve à faire à cette dernière affirmation. Elle a trait à la propriété de maximalité dont nous avons parlé dans la section 3.5.2.1.2. Supposons que nous utilisons une couverture de Σ composée de cliques maximales, et qu'aucune erreur ne soit autorisée entre un modèle et ses occurrences. Étant donné un ensemble de chaînes $s1$,$s2$,...,sN et un modèle de longueur k qui est EL-présent avec 0 erreurs dans q de ces chaînes, nous ne pouvons garantir que son ensemble d'occurrences soit une clique maximale dans le sens où il ne contiendrait l'ensemble

Supposons que nous travaillons avec la définition 3.5.3 de similarité. Soient $\Sigma = \{a, b\}$, $q = 4$ et $e = 2$ (au plus deux substitutions sont autorisées entre un modèle et ses occurrences). Soient les quatre chaînes :

$s1 = aaa$
$s2 = aaa$
$s3 = aaa$
$s4 = bbb$.

Alors les modèles *aab*, *aba*, *baa*, *abb*, *bba* et *bbb* sont tous des modèles de longueur 3 valables d'après la définition 3.5.3 (ce sont les seuls) mais le centre de gravité de l'ensemble d'occurrences de ces modèles est le mot *aaa* qui n'est pas un modèle valide (il n'est pas présent sur la dernière chaîne).

Figure 3.31: Un exemple indiquant pourquoi, étant donné un ensemble de mots similaires (i.e. d'occurrences), il n'existe pas toujours un modèle qui représente un centre de gravité ou un mot de Steiner pour cet ensemble.

d'occurrences d'aucun autre modèle de même longueur. L'exemple de la figure 3.23 illustre ce point à nouveau puisque $\{I, L\}\{I, L\}$ et $\{L, M\}\{I, L\}$ sont tous deux des modèles de longueur 2 qui sont valables suivant la définition 3.5.8, mais en termes d'ensembles d'occurrences, le premier est complètement inclus dans le second et ne peut donc être une clique maximale. Notons qu'il s'agit ici d'une non-maximalité par rapport aux chaînes qui sont comparées et non intrinsèque (voir définition 3.5.6). En termes pratiques cela signifie cependant simplement que, si ce sont les cliques maximales qui nous intéressent, il faut pendant tout le processus d'identification des groupes de mots similaires d'un ensemble de chaînes, tester à chaque étape que les modèles obtenus soient maximaux en tant qu'ensembles d'occurrences. À part ce fait, nous pouvons considérer alors que les cliques maximales (et les classes d'équivalences) sont aussi des modèles, c'est-à-dire des produits d'ensembles d'une couverture de Σ.

Qu'en est-il des deux autres mesures directes, s'appuyant soit sur l'optimisation d'une fonction de score, soit sur des calculs statistiques? La première présente un point commun important avec les mesures indirectes et avec celle établie à partir d'une relation R. En effet, la définition de ressemblance entre mots qu'elle utilise conduit aussi à une construction par récurrence des blocs de similarité. Par ailleurs, l'idée d'une optimisation reste dissimulée derrière les mesures qui utilisent des modèles mais, d'une part, la façon de calculer les scores n'est plus la même et, d'autre part, les modèles ne sont pas nécessairement optimaux. Vérifient-ils une quelconque propriété de maximalité (ou de minimalité)? Le plus intéressant sans doute serait de pouvoir affirmer que les modèles sont des centres de gravité [Waterman, 1995] ou des mots de Steiner [Gusfield, 1995] pour leurs occurrences. Cela serait le cas si la somme des mesures de ressemblance entre un modèle et chacune de ses occurrences était maximale ou minimale. Toutefois cela n'est pas vrai pour tous les modèles. Il n'est même pas possible d'affirmer que, étant donné un ensemble de mots similaires (i.e. un ensemble d'occurrences), il existe toujours au moins un modèle représentant un centre de gravité ou un mot de Steiner. L'exemple de la figure 3.31 illustre ce point. Observons cependant que, au moins en ce qui concerne les définitions de ressemblance 3.5.3 et 3.5.8 présentées plus haut, les ensembles d'occurrences trouvés par les modèles peuvent être plus consistants (voir la section 3.3.2) que les ensembles de mots localement similaires repérés par programmation dynamique (voir la section 3.4.3.1). En effet, dans

le premier cas la distance entre deux occurrences d'un même modèle est toujours d'au plus 2e, alors que dans le second la distance entre deux mots appartenant à un ensemble plus grand de mots localement similaires peut être faible puisque la similarité a pour base un score SP, c'est-à-dire la somme de tous les scores deux à deux.

Enfin, existe-t-il un point commun du même genre entre la définition 3.4.11 de similarité établie à partir de critères statistiques et les quatre dernières utilisant des modèles? En général non, bien que cela puisse dépendre la formule utilisée pour estimer la similarité d'un groupe de mots de manière statistique. Les deux mesures sont très différentes par contre dans la façon d'obtenir les blocs de similarité ainsi que nous l'avons vu. En particulier, les méthodes stochastiques ne construisent pas les blocs par récurrence et ne sont pas exactes.

3.5.5 Diagramme résumant toutes les définitions de ressemblances

Le diagramme de la figure 3.32 montre les rapports entre toutes ces mesures de ressemblance.

Figure 3.32: Diagramme résumant toutes les définitions de ressemblance.

Chapitre 4

Les méthodes directes de comparaison

Experiment to me
Is every one I meet
If it contain a Kernel?
The Figure of a Nut

Presents upon a Tree
Equally plausibly,
But Meat within, is requisite
To Squirrels, and to Me.

Poème #1719. Emily Dickinson.
The Complete Poems of Emily Dickinson.
Thomas H. Johnson (Ed.). Faber and Faber, 1975.

Ce chapitre a pour but de présenter l'approche directe de la comparaison des chaînes macromoléculaires. Il se trouve que cette approche englobe presque toutes les méthodes d'analyse hors celles que nous avons nous-mêmes élaborées et qui sont introduites dans le chapitre suivant. C'est donc une revue des méthodes développées par d'autres que nous donnons ici.

Ces méthodes se divisent en deux grands groupes, celles qui comparent les chaînes dans leur totalité, et celles qui effectuent des comparaisons locales. Mais nous pouvons en réaliser également une autre classification, comprenant les méthodes ayant pour base l'optimisation d'une fonction de score, les méthodes combinatoires exactes et enfin celles qui recherchent une optimisation stochastique et qui sont les plus proches des méthodes indirectes.

Le plan de ce chapitre suit ces divisions naturelles. Dans une première partie nous présentons donc les méthodes globales, puis dans une seconde celles locales. Les méthodes globales sont en général toutes basées sur des optimisations d'une certaine fonction de score. Les locales peuvent l'être, mais elles concernent également l'approche combinatoire exacte et les optimisations stochastiques. Cette seconde partie se divise ainsi en trois sous-parties.

Notre intention n'est pas de détailler chacune de ces approches. La personne intéressée peut le faire en consultant les articles cités tout au long du chapitre. Notre objectif est simplement d'essayer de dégager les idées principales qui se cachent sous une grande variété apparente de méthodes et d'introduire certaines difficultés inhérentes au problème de la comparaison multiple de chaînes.

La première de ces difficultés est d'ordre conceptuel. Nous avons vu en effet que comparer plus de deux objets à la fois est une opération pour laquelle nous ne disposons d'aucune définition 'naturelle'. Commencer par établir une telle définition est donc une tâche importante qui n'est pas toujours faite de manière très précise. Certaines définitions existent pourtant, nous les avons présentées au chapitre précédent, mais elles mènent souvent à des méthodes de calculs exacts qui sont impraticables. La seconde difficulté à laquelle nous sommes confrontés lorsque nous réalisons des comparaisons multiples est ainsi d'origine algorithmique et est essentiellement liée à la nature combinatoire du problème. Nous verrons que cela a conduit à l'élaboration de nombreuses heuristiques pour comparer un grand nombre d'objets simultanément d'une manière qui soit efficace en pratique.

4.1 Approche globale

4.1.1 Base de départ : le deux à deux

4.1.1.1 Principe — la programmation dynamique

La base de départ de toute comparaison d'un ensemble de chaînes entre elles est la comparaison de celles-ci deux à deux ainsi que nous l'avons vu au chapitre précédent. La mesure de ressemblance adoptée concerne l'optimisation d'une fonction de score associée à un alignement des deux chaînes où le score peut indiquer soit une similarité entre chaînes, soit une dissimilarité ou encore une distance.

Étant donné une telle fonction, la méthode classique et exacte pour trouver le meilleur alignement entre 2 chaînes $a_1...a_n$ et $b_1...b_m$, c'est-à-dire celui qui optimise la valeur de cette fonction, est alors la programmation dynamique [Needleman and Wunsch, 1970] [Sellers, 1974] [Wagner and Fischer, 1974]. Il ne s'agit de rien d'autre que d'une méthode algorithmique de calcul qui reflète la nature récurrente de la notion de ressemblance choisie.

Cette méthode consiste en effet à construire par induction un tableau $\mathcal{D}$ de dimension proportionnelle au produit des longueurs des 2 chaînes où la valeur de la cellule $\mathcal{D}(i,j)$ pour $1 \leq i \leq n$ et $1 \leq j \leq m$ indique le score de l'alignement optimal des chaînes jusqu'aux positions i et j respectivement et peut être déduite des valeurs de trois des cellules adjacentes du tableau, suivant en cela la définition 3.2.20. Nous avons ainsi (voir figure 4.1) :

$$\mathcal{D}(i,j) = max/min\{\mathcal{D}(i-1,j-1) + score\begin{pmatrix} a_i \\ b_j \end{pmatrix}, \mathcal{D}(i-1,j) + w_\lambda, \mathcal{D}(i,j-1) + w_\lambda\}$$

où w_λ est la valeur attribuée à un trou et la valeur de $score\begin{pmatrix} a_i \\ b_j \end{pmatrix}$ est donnée par une matrice, ou est obtenue à partir d'une relation ou d'une couverture (voir définition 3.2.19). Notons que, dans le cas d'une mesure de similarité, la valeur de w_λ est au plus égale à la plus petite valeur de substitution d'un symbole par un autre (en général, elle est donc négative).

Les valeurs frontières du tableau sont définies par :

$$\mathcal{D}(i,0) = i \times w_\lambda,\ \mathcal{D}(0,j) = j \times w_\lambda.$$

Le calcul du score optimal d'un alignement de deux chaînes peut également être vu en termes d'un parcours dans un graphe orienté dont les nœuds représentent les paires (i,j) pour $0 \leq i \leq n$ et $0 \leq j \leq m$ et où le nœud $(i-1,j-1)$ est lié par une arête à chacun des trois nœuds $(i-1,j)$, $(i,j-1)$ et (i,j) lorsque ceux-ci sont définis. Le poids des deux premières arêtes est égal à la valeur affectée à un trou et le poids de la dernière à celui de la substitution de a_i par

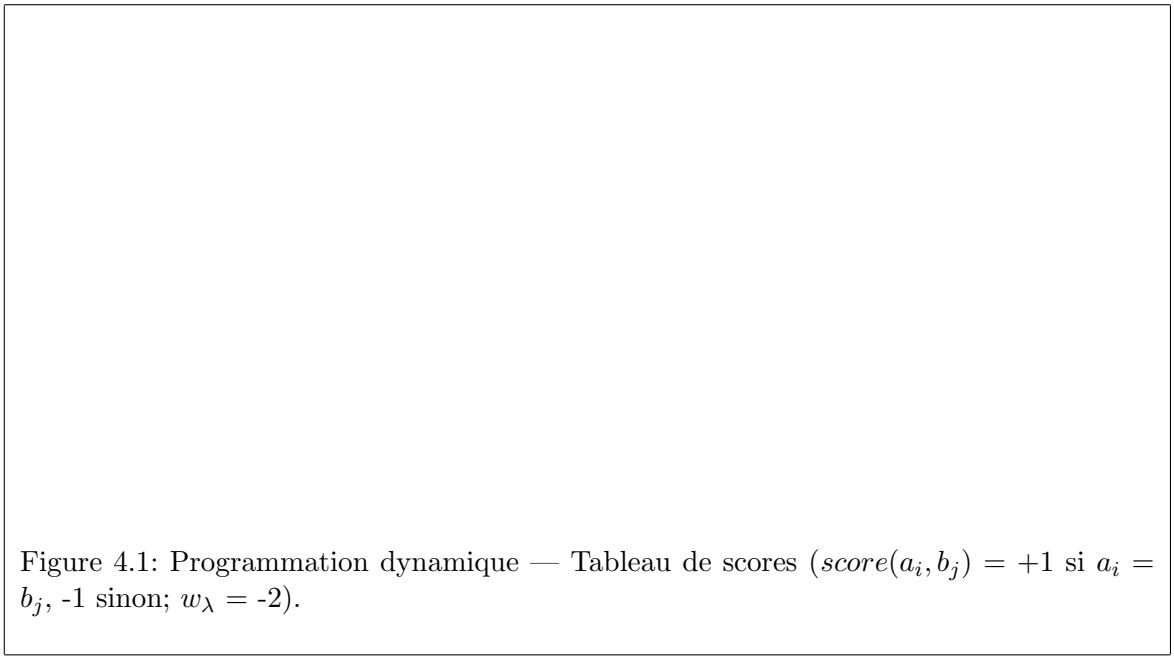


Figure 4.1: Programmation dynamique — Tableau de scores ($score(a_i, b_j) = +1$ si $a_i = b_j$, -1 sinon; w_λ = -2).

b_j. Trouver le score optimal correspond alors à trouver un chemin de plus grand poids dans ce graphe (ou de plus petit dans le cas d'une mesure de dissimilarité ou d'une distance) allant du nœud source (0,0) jusqu'au nœud puits (n, m) (voir figure 4.2). Ce chemin correspond alors à celui d'un 'meilleur' alignement des deux chaînes.

Les deux approches sont souvent combinées, et un tableau de pointeurs est construit en même temps que le tableau des scores (voir figure 4.3). Ce second tableau représente en fait le graphe antérieurement donné pour lequel chaque arête se trouve maintenant orientée en sens inverse. Grâce à lui, un meilleur chemin de la source jusqu'au puits peut être retrouvé à partir du calcul du meilleur score, et ainsi un meilleur alignement de $a_1...a_n$ et $b_1...b_m$, sans avoir besoin de refaire tous les calculs à l'envers.

Lorsqu'une des chaînes est beaucoup plus courte que l'autre, il est clair que c'est uniquement le meilleur alignement de cette chaîne plus courte avec un mot de l'autre qui est recherché. La formule de récurrence demeure la même, seules changent les valeurs des cellules situées à une des frontières du tableau. En plaçant la chaîne plus longue en haut de celui-ci, on a en effet $D(i, 0) = 0$ pour $1 \leq i \leq n$. L'alignement optimal est alors obtenu en partant d'une cellule de la dernière ligne du tableau ayant la valeur maximale (ou minimale) et en remontant le tableau des pointeurs jusqu'à la première ligne. Cette variante consiste à réaliser un meilleur ajustement de la chaîne courte avec celle plus longue (en anglais, 'Best fit').

Observons enfin que l'alignement optimal ne représente pas toujours celui qui est 'correct' biologiquement, ce dernier correspondant plutôt à un alignement ayant un score élevé proche mais non égal au maximum (minimum). Des variantes de la programmation dynamique existent alors qui calculent tous les alignements globaux dont le score se trouve à une certaine distance du score optimal [Waterman, 1983] [Waterman and Byers, 1985] [Zuker, 1991].

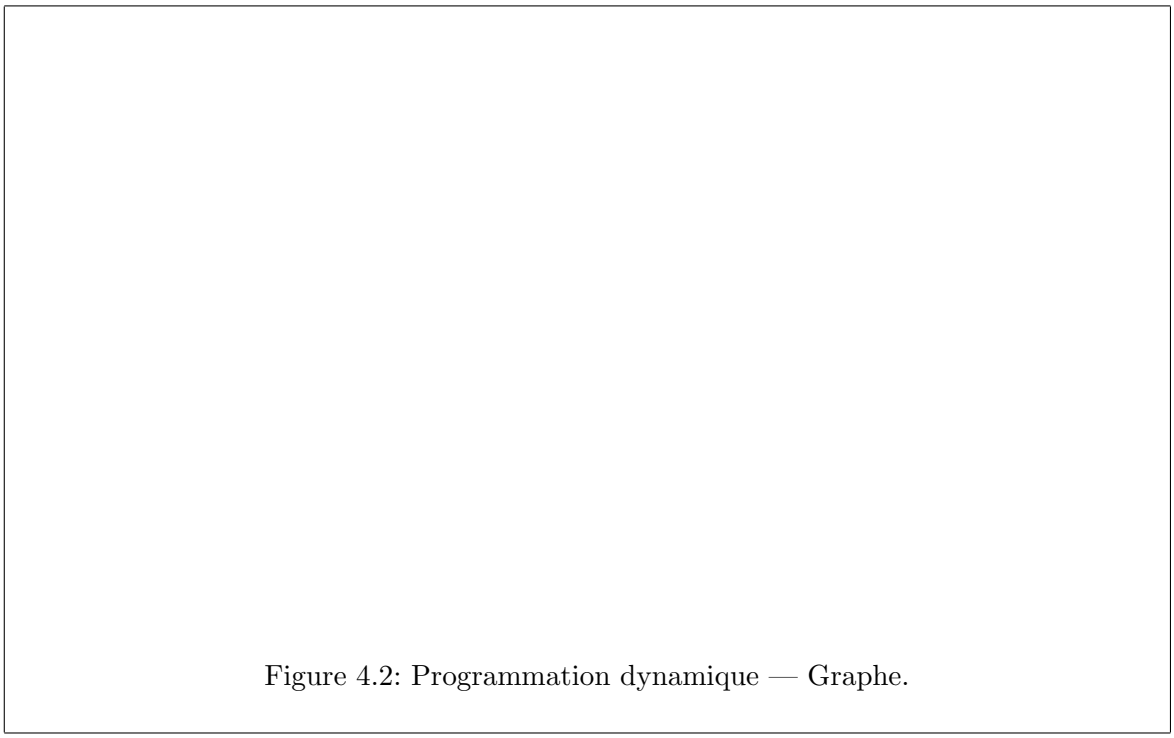


Figure 4.2: Programmation dynamique — Graphe.

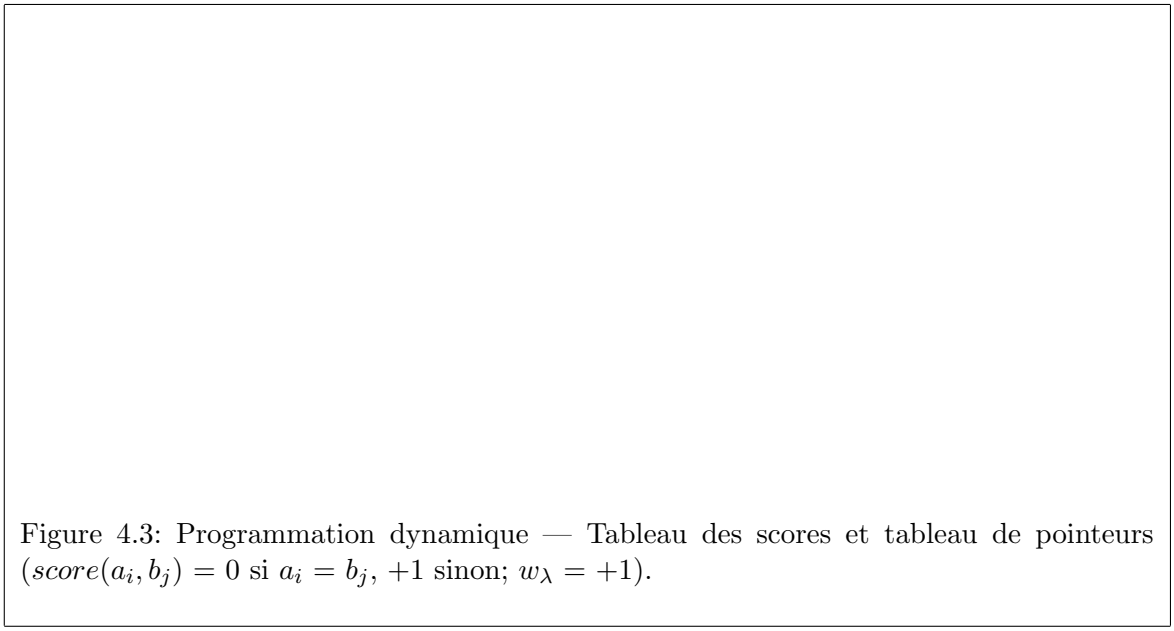


Figure 4.3: Programmation dynamique — Tableau des scores et tableau de pointeurs ($score(a_i, b_j) = 0$ si $a_i = b_j$, $+1$ sinon; $w_\lambda = +1$).

4.1.1.2 Une discussion sur la complexité

4.1.1.2.1 Temps

La complexité en temps de l'algorithme de programmation dynamique est $O(n.m)$. Il a été montré [Wong and Chandra, 1976] que ce temps est aussi $\Omega(n.m)$ pour des alphabets de taille infinie, lorsque seules des comparaisons du type 'égal-ou-différent' peuvent être réalisées et que la complexité de l'algorithme est caractérisée exclusivement en termes de la taille de l'entrée. Cette borne inférieure est obtenue même lorsque le système de score mesure un cas spécial de la distance d'édition entre les chaînes [Aho *et al.*, 1974] où seules les délétions et insertions sont autorisées. Ce que nous recherchons alors est une plus longue sous-séquence commune à celles-ci ($s' = c_1...c_p$ est appelée une sous-séquence d'une chaîne $s = a_1...a_n$ si s' peut être obtenu de s en supprimant certains symboles de s).

Lorsque l'alphabet est de taille finie, et que les scores attribués aux paires représentent une distance d'édition ou ont des poids qui sont des entiers multiples d'un même nombre réel r, il est possible, en utilisant la technique appelée des Quatre Russes, de calculer le score d'un alignement global des deux chaînes $s1 = a_1...a_n$ et $s2 = b_1...b_m$ en un temps $O((n.m)/\log n)$ [Masek and Paterson, 1983] si l'on suppose, sans perte de généralité, que $n \leq m$.

La complexité en temps de l'algorithme de programmation dynamique peut être également caractérisée en termes d'autres paramètres que la seule taille de l'entrée. Lorsque nous cherchons par exemple à déterminer la distance d'édition entre $s1$ et $s2$, un paramètre d peut être défini qui représente une borne supérieure estimée de cette distance. Des variantes de l'algorithme de programmation dynamique ont alors été élaborées [Fickett, 1984] [Nakatsu *et al.*, 1982] [Spouge, 1989] [Spouge, 1991] [Ukkonen, 1985a] [Ukkonen, 1985b] qui calculent le score optimal global des deux chaînes $s1$ et $s2$ en un temps $O(n.d)$. La technique consiste à limiter le volume de l'espace à explorer, c'est-à-dire le nombre de cellules du tableau $\mathcal{D}$ qu'il faut effectivement calculer (ou de nœuds du graphe par lesquels il faut passer). L'idée est la suivante. Rappelons que nous raisonnons dans cet exemple en termes de distance plutôt que de similarité, et utilisons la double représentation tableau de scores-graphe (i.e. tableau de pointeurs). Lorsque les deux chaînes $s1$ et $s2$ ne sont pas trop dissemblables, un chemin optimal dans le graphe se situe en général proche de la diagonale principale du tableau, en fait il est entièrement contenu dans une bande de largeur $2d$ centrée sur cette diagonale (voir figure 4.4). Seules les cellules situées à l'intérieur de cette bande ont alors besoin d'être calculées.

Il est possible d'être encore plus économe que cela en travaillant non avec une seule estimation — celle du paramètre d — mais avec deux. La seconde représente non plus une borne supérieure de la distance $\mathcal{D}(n, m)$ entre $a_1...a_n$ et $b_1...b_m$ mais une borne inférieure κ de cette distance [Spouge, 1989] [Spouge, 1991]. Cette borne est en fait estimée pour chaque nœud x du graphe, et indique la valeur minimale que peut avoir la longueur d'un chemin passant par $x = (i, j)$ et allant de la source au puits. Elle est donc notée κ_x(source,puits) = κ(source,x) + κ(x,puits). Elle permet ainsi d'éventuellement réduire encore plus le nombre de cellules à calculer à l'intérieur de la bande, cela à la fois de manière directe (si pour un x donné à l'intérieur de celle-ci on a κ_x(source,puits) $> d$), et de manière indirecte puisque l'estimation de κ_x(source,puits) pour tout x permet d'établir un ordre parmi les cellules du tableau. Celles ayant une plus petite valeur estimée de κ_x sont alors calculées en priorité et la cellule $\mathcal{D}(n, m)$ peut être atteinte avant d'avoir eu à déterminer la valeur des autres cellules situées dans la bande de largeur $2d$. Aussi bien d que κ_x peuvent être estimés de façon statique, c'est-à-dire une fois pour toutes, ou dynamique. Dans ce second cas et en ce qui concerne κ_x, on a κ_x(source,puits) = $\mathcal{D}(i, j)$ + κ(x,puits) (rappelons que (i, j) représente le nœud x et $\mathcal{D}(i, j)$ la distance entre $a_1...a_i$ et $b_1...b_j$). Une estimation dynamique de la valeur de d dépend elle aussi du chemin

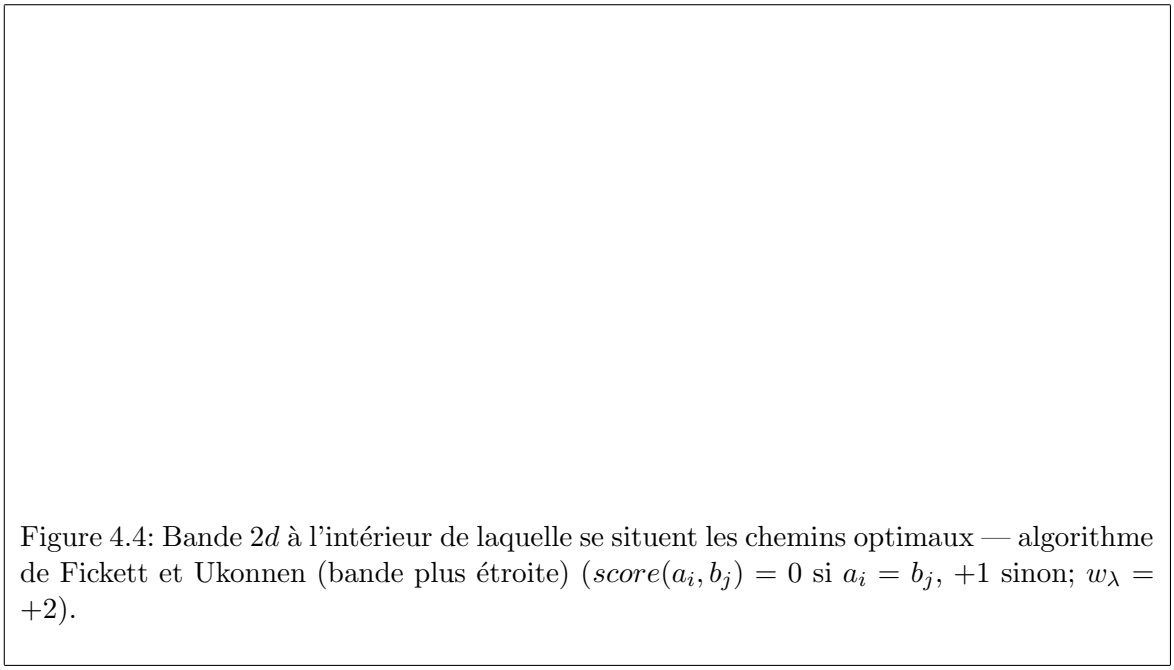


Figure 4.4: Bande $2d$ à l'intérieur de laquelle se situent les chemins optimaux — algorithme de Fickett et Ukonnen (bande plus étroite) ($score(a_i, b_j) = 0$ si $a_i = b_j$, $+1$ sinon; $w_\lambda = +2$).

déjà parcouru et on a alors d_x(source,puits) = $\mathcal{D}(i,j) + \mu(x$,puits) où $\mu(x$,puits) est une borne supérieure pour la distance séparant $a_{i+1}...a_n$ et $b_{j+1}...b_m$. L'estimation dynamique des bornes inférieures et supérieures de la distance $\mathcal{D}(n,m)$ permet bien sûr de mieux réduire l'espace à explorer mais elle représente elle-même un certain coût en espace et en temps. Ce coût est en général d'autant plus élevé que l'estimation est bonne et un équilibre entre réduire l'espace au mieux et dépenser trop de temps à estimer les bornes doit donc être trouvé. La différence essentielle entre les versions mentionnées ci-dessus porte justement sur la manière de réaliser cette estimation de d et κ. Ficket par exemple n'estime que la valeur de d et le fait de manière statique. De même qu'Ukkonen, il 'devine' en fait cette valeur, et éventuellement la revoit à la hausse si dans une première application de l'algorithme la cellule (n,m) du tableau n'est pas atteinte (parce que $\mathcal{D}(n,m)$ est supérieur à la valeur 'devinée'). Spouge estime de manière dynamique aussi bien la valeur de d_x(source,puits) que celle de κ_x(source,puits) pour tout nœud $x = (i,j)$ du graphe d'après le nombre maximal de substitutions, délétions et insertions (estimation de d_x) ou le nombre minimal de délétions et insertions (estimation de κ_x) nécessaires pour aligner $a_{i+1}...a_n$ et $b_{j+1}...b_m$.

Une réduction de l'espace à explorer peut être obtenue avec des mesures de ressemblance entre chaînes autres que la distance d'édition. Celles-ci peuvent ainsi concerner des mesures de similarité ou de dissimilarité quelconques si le poids attribué à un trou est constant, ou bien une forme spéciale de la distance d'édition déjà mentionnée où seules des délétions et insertions sont autorisées entre $a_1...a_n$ et $b_1...b_m$, c'est-à-dire lorsque l'on recherche en fait une sous-séquence maximale commune aux deux chaînes. Dans ce dernier cas, Myers obtient un algorithme initialement en $O(n.d)$ où d mesure cette fois la taille de la plus petite suite d'opérations de délétions et insertions nécessaires pour passer de $s1 = a_1...a_n$ à $s2 = b_1...b_m$ [Myers, 1986], puis en $O(n.p)$ où p est le nombre de délétions dans cette suite et est égal à $\frac{1}{2}d - \frac{1}{2}(m-n)$ [Wu *et al.*, 1990]. Comme précédemment pour Spouge, on ne parcourt pas la matrice de manière strictement linéaire ligne par ligne et colonne par colonne, mais on serpente à l'intérieur d'une bande de largeur $2d$ ou $2p + (n-m)$, l'algorithme calculant en

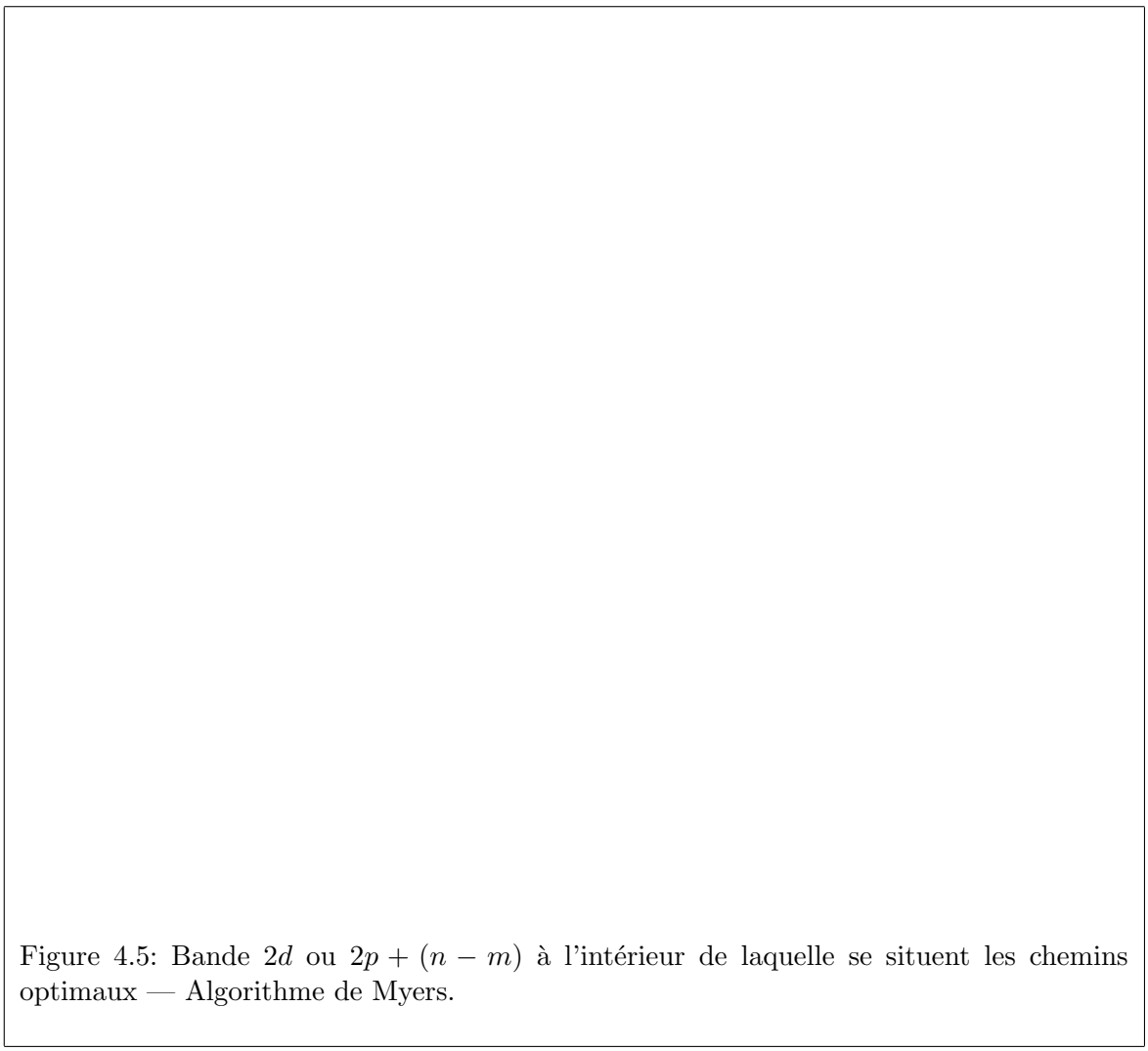


Figure 4.5: Bande $2d$ ou $2p + (n - m)$ à l'intérieur de laquelle se situent les chemins optimaux — Algorithme de Myers.

effet successivement les cellules de la bande jusqu'à une certaine frontière correspondant à un nombre de trous (ou de délétions) allant de 0 à d (ou p) (voir la figure 4.5).

Toutes ces économies qui peuvent être réalisées valent aussi bien pour le calcul du seul score d'un alignement optimal que pour l'obtention d'un tel alignement. Il est très important d'observer toutefois que ces économies dépendent des données. Dans la pire des hypothèses, c'est-à-dire en termes d'une distance d'édition si $d = m$ (les chaînes sont totalement 'différentes'), la complexité demeure en $O(n.m)$. Signalons que toutes ces techniques d'exploration exhaustives avec bornes correspondent à ce qui est appelé en anglais du 'branch-and-bound'.

Enfin, toutes les complexités présentées jusqu'à maintenant ne concernent que le cas où le score attribué à un trou est une fonction constante ou linéaire de sa longueur. Lorsqu'une fonction linéaire par t morceaux est utilisée, la complexité devient $O(n^2.t)$ si l'on suppose que $m = n$ [Gotoh, 1982] [Miller and Myers, 1988], elle est en $O(n^2.\log n)$ pour une fonction concave [Galil and Giancarlo, 1989] [Miller and Myers, 1988] [Waterman, 1984a] et en $O(n^3)$ pour une fonction quelconque [Waterman *et al.*, 1976].

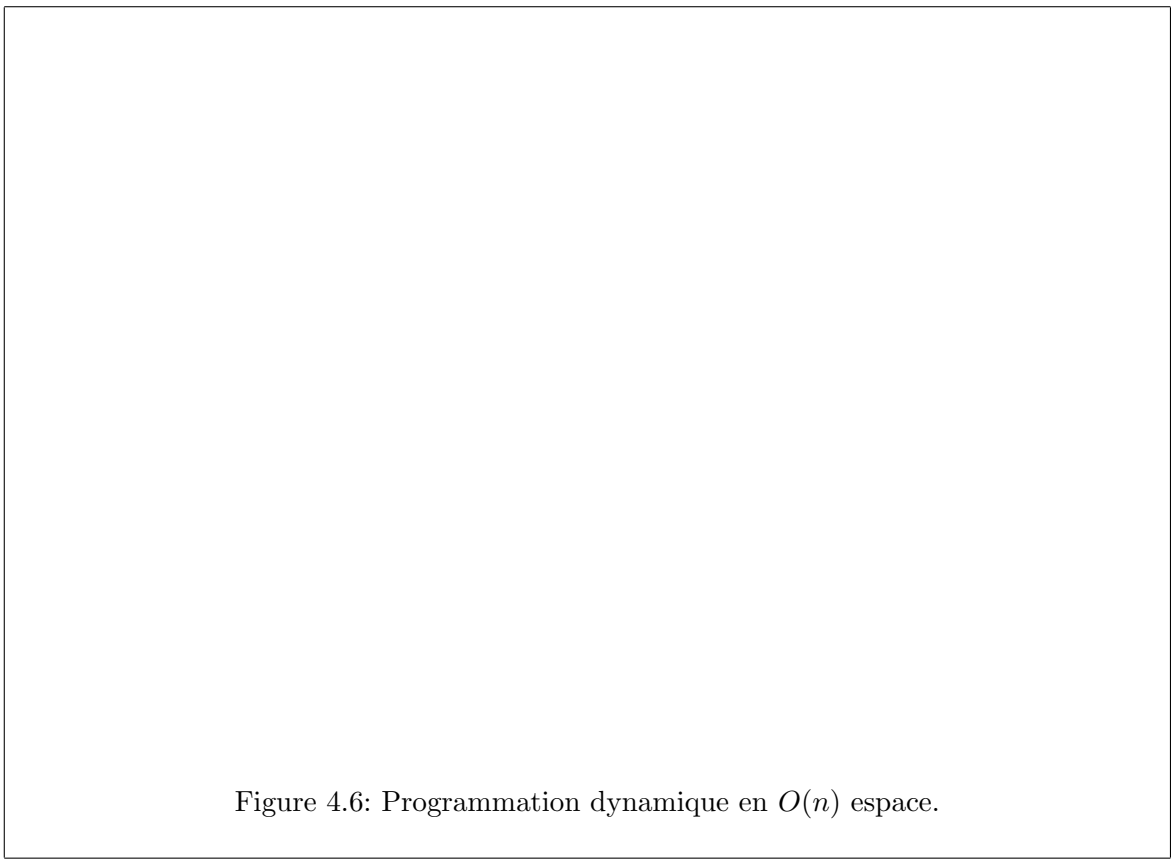


Figure 4.6: Programmation dynamique en $O(n)$ espace.

4.1.1.2.2 Espace

La complexité en espace de l'algorithme de programmation dynamique pour le cas global est en $O(min(n, m))$ aussi bien pour le calcul du meilleur score que pour la détermination d'un meilleur alignement (chemin dans le graphe).

En ce qui concerne le calcul du meilleur score, il est en effet assez intuitif de voir que nous n'avons besoin de connaître que les valeurs des cellules de la ligne $i-1$ (colonne $j-1$) pour calculer celles de la ligne i (colonne j). Il est moins facile de comprendre pourquoi trouver un meilleur chemin peut également être réalisé en $O(min(n, m))$ espace. L'idée est de diviser successivement le problème initial en deux sous-problèmes de taille plus petite impliquant l'alignement pour tout j et pour $n \leq m$ de $a_1...a_{\frac{n}{2}}$ avec $b_1...b_j$ et de $a_{\frac{n}{2}+1}...a_n$ avec $b_{j+1}...b_m$ et pour chaque sous-problème de déterminer un point intermédiaire j_{opt} optimal [Myers and Miller, 1988]. La liste de ces j_{opt} constitue alors un chemin optimal recherché (voir figure 4.6).

4.1.1.2.3 Commentaire

Si nous nous sommes attardés sur cette question de la complexité de l'algorithme de programmation dynamique c'est qu'elle va devenir cruciale lorsque nous passerons dès la section suivante au cas d'un alignement ou d'une comparaison multiple de chaînes. Nous allons voir en effet qu'une extension directe de la programmation dynamique au cas de plus de deux chaînes résulte en un algorithme qui est exponentiel dans le nombre de celles-ci qu'il y a à comparer. Ce

qu'il est important à noter est que ce phénomène est intrinsèque au problème lorsque celui-ci est traité dans sa plus grande généralité et il a été montré que, dans ce cas, ce problème est en fait NP-complet [Garey and Johnson, 1979]. Les gains qui peuvent être obtenus malgré tout vont donc toujours dépendre de la nature particulière des données à traiter.

Nous verrons plus tard que le cas des comparaisons locales est différent. Une application directe de la programmation dynamique conduit à une explosion combinatoire des opérations à effectuer qui est pire en pratique que dans le cas global dans la mesure où plus aucune économie n'est possible. Cependant des méthodes indirectes de comparaison qui sont également récursives dans leur principe permettent, sous certaines conditions, d'obtenir des temps théoriques de calcul bien moindres tout en demeurant exacts et exhaustifs.

4.1.2 Le multiple exact

4.1.2.1 Du deux à deux au multiple : rappel sur les scores

Commençons par rappeler que les scores d'un alignement multiple peuvent être calculés essentiellement de deux façons. Dans la première, le score d'un alignement de plus de deux chaînes est la somme des scores de tous les alignements deux à deux qu'il induit (voir la figure 3.19). On parle dans ce cas de scores SP (voir section 3.4.1.2). La seconde utilise un arbre phylogénétique pour obtenir un alignement des chaînes et le score de cet alignement est la somme de tous les alignements deux à deux associés à chaque branche de l'arbre et qui sont induits par l'alignement multiple (voir la figure 3.21). En général, l'arbre est connu au départ (il est supposé avoir été déterminé par d'autres moyens qu'une analyse des chaînes macromoléculaires) et il s'agit simplement de trouver les chaînes ancestrales situées aux nœuds internes de l'arbre (les chaînes à aligner étant placées aux feuilles) (voir section 3.4.1.3). Des algorithmes exacts n'existent que dans le cas où l'arbre est donné *a priori* et nous ne parlerons pas du tout du cas où celui-ci doit être construit en même temps que l'alignement [Vingron and von Haesler, 1994].

4.1.2.2 Algorithmes exacts utilisant des scores SP

4.1.2.2.1 Extension directe de la programmation dynamique — problème de la complexité

Les algorithmes permettant de construire des alignements globaux optimaux sur la base de scores SP sont des extensions directes de la programmation dynamique à plus de deux dimensions. Ils ont été élaborés par Murata [Murata, 1990] [Murata *et al.*, 1985] et Gotoh [Gotoh, 1986] et consistent à calculer par induction les valeurs d'un tableau $\mathcal{D}$ à N dimensions et non plus à deux, où N est le nombre de chaînes à comparer ou aligner et $\mathcal{D}(i_1, ..., i_N)$ indique le score d'un alignement optimal des chaînes $s_1, ..., s_N$ jusqu'aux positions $i_1, ..., i_N$ respectivement. La valeur de la cellule $(i_1, ..., i_N)$ du tableau $\mathcal{D}$ peut être obtenue, de la même façon que pour le deux à deux, directement des valeurs des $2^N - 1$ cellules adjacentes (voir figure 4.7 pour $N = 3$). À nouveau ici, le problème peut également être vu comme celui de trouver un chemin de plus grand (ou de plus petit) poids dans un graphe orienté.

La principale difficulté avec cette approche est, ainsi que nous l'avons mentionné, la complexité en temps et en espace d'un tel algorithme. Celle-ci est en effet en $O(n^N)$ où n est la longueur moyenne des chaînes ($O(n^{N-1})$ espace lorsque seul le meilleur score est recherché). Il est clair que la méthode n'est envisageable que pour des valeurs petites de N, en pratique elle est en fait limitée à au plus 3 chaînes.

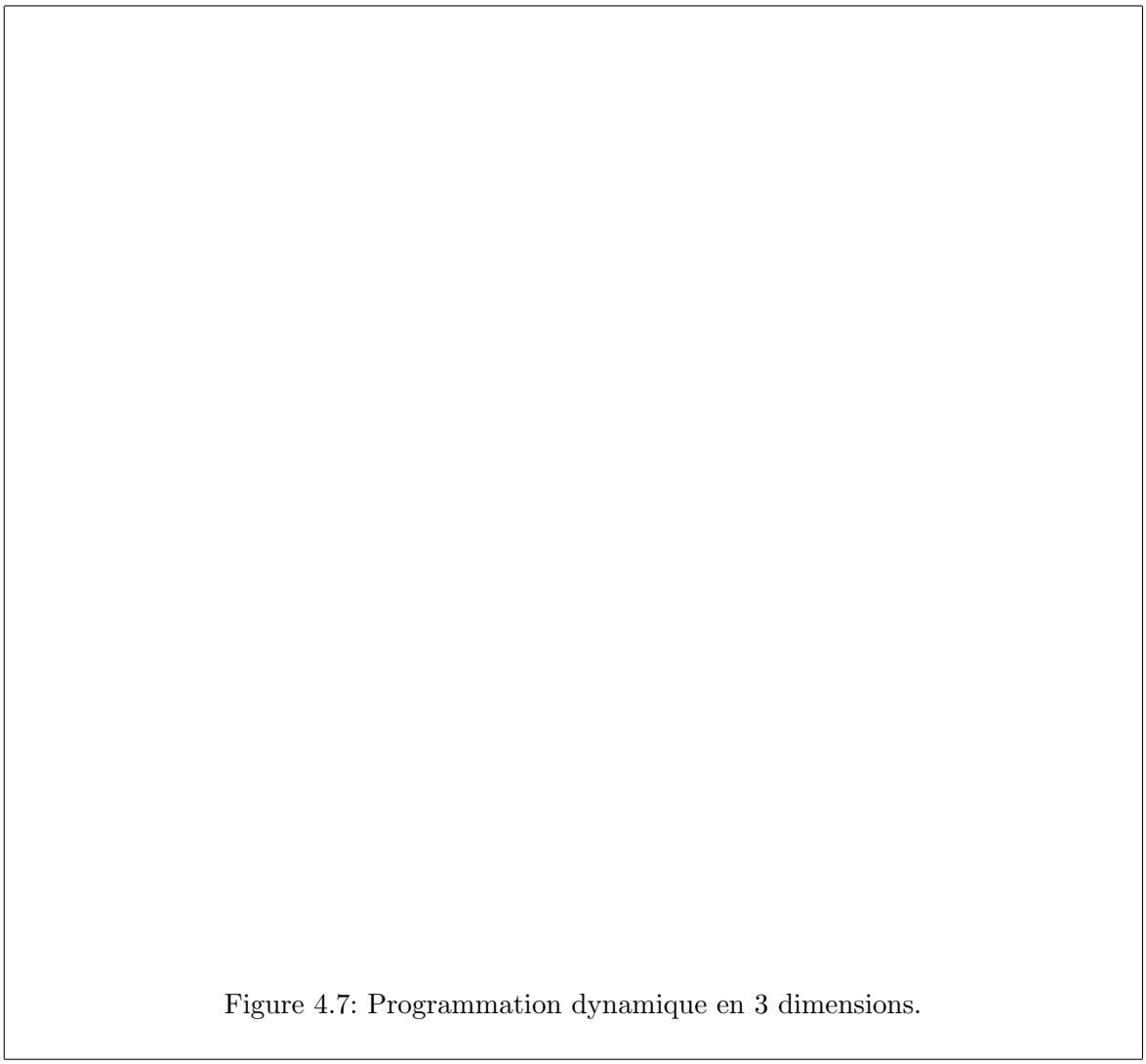


Figure 4.7: Programmation dynamique en 3 dimensions.

4.1.2.2.2 Économies possibles

Comme dans le cas du deux à deux, il est cependant possible d'essayer de limiter le volume de l'espace à explorer [Carrillo and Lipman, 1988] [Gupta *et al.*, 1995] [Lipman *et al.*, 1989], c'est-à-dire le nombre de cellules du tableau à N dimensions qu'il faut calculer. L'idée de base est la même que précédemment dans la mesure où il s'agit d'estimer certaines bornes, inférieures ou supérieures, du score recherché, mais elle exploite cette fois la nature des scores utilisés pour mesurer la qualité d'un alignement multiple. Comme auparavant, raisonnons en termes de distance plutôt que d'une mesure de similarité. Supposons qu'un alignement multiple quelconque de N chaînes ait pu être déterminé, par une heuristique par exemple. Notons son score $score(\mathcal{H})$. Si $score(\mathcal{O})$ est le score d'un alignement optimal des N chaînes, alors nécessairement $score(\mathcal{H}) \geq score(\mathcal{O})$. Mais par définition des scores SP, $score(\mathcal{O})$ est lui-même égal à la somme des scores de toutes les projections de l'alignement multiple $\mathcal{O}$ sur les N chaînes prises deux à deux, c'est-à-dire :

$$score(\mathcal{O}) = \sum_{1 \leq i < j \leq N} score(\mathcal{O}_{ij})$$

où $\mathcal{O}_{ij}$ dénote la projection de l'alignement $\mathcal{O}$ sur les chaînes si et sj. Par ailleurs, chacun de ces scores entre paires de chaînes si, sj est supérieur ou égal à la distance entre si et sj. On a donc :

$$score(\mathcal{H}) \geq score(\mathcal{O}) = \sum_{1 \leq i < j \leq N} score(\mathcal{O}_{ij}) \geq \sum_{1 \leq i < j \leq N} distance(si, sj)$$

où $distance(si, sj)$ mesure la distance entre si et sj. En distinguant l'une de ces paires, on obtient :

$$\sum_{\substack{1 \leq i < j \leq N \\ i \neq p, j \neq q}} score(\mathcal{O}_{ij}) \geq score(\mathcal{O}_{pq}) - distance(sp, sq) \; + \sum_{1 \leq i < j \leq N} distance(si, sj).$$

Ceci implique que :

$$score(\mathcal{O}_{pq}) \leq score(\mathcal{H}) + distance(sp, sq) - \sum_{1 \leq i < j \leq N} distance(si, sj).$$

En d'autres termes, le score de la projection d'un alignement optimal $\mathcal{O}$ de N chaînes sur une paire donnée de celles-ci est inférieur ou égal au score d'un alignement quelconque $\mathcal{H}$ des N chaînes moins la somme des distances entre toutes les paires restantes. En pré-calculant toutes ces distances deux à deux et en déterminant le score d'un alignement quelconque $\mathcal{H}$ des N chaînes, il est ainsi possible d'obtenir des bornes supérieures pour les scores des projections d'un alignement optimal sur chacune des paires de chaînes de l'ensemble de départ. De telles bornes limitent le nombre de cellules qu'il faut calculer sur chaque tableau projeté à deux dimensions, et donc limite le nombre de cellules qu'il faut calculer dans le tableau à N dimensions.

La différence fondamentale entre les algorithmes de Carrillo et Lipman d'un côté et de Lipman *et al.* et Gupta *et al.* d'un autre est que ces derniers peuvent dériver les bornes pour les scores des paires directement de l'alignement $\mathcal{H}$ et non de l'inégalité donnée ci-dessus. Cette borne est alors, pour une paire de chaînes donnée sp et sq, non pas le score de $\mathcal{H}$ moins la somme des distances entre toutes les autres paires, mais directement le score de la projection de $\mathcal{H}$ sur sp et sq. Ce score peut être plus petit que la borne estimée par Carrillo et peut ainsi mieux réduire l'espace à explorer, mais il ne garantit pas qu'un score optimal sera obtenu à la fin. Par ailleurs, Gupta *et al.* calculent aussi une borne inférieure aux scores des paires comme il était fait dans le cas du deux à deux (estimation de κ_x). Enfin, les algorithmes de Altschul et de

Gupta différent également de celui de Carrillo dans la mesure où les deux premiers pondèrent la contribution de chaque chaîne à l'alignement final. L'idée est de compenser l'influence trop forte que pourrait avoir la présence dans un alignement de plusieurs chaînes presque identiques.

Quel que soit l'algorithme utilisé parmi ceux mentionnés ici, la complexité de pire cas est la même que celle d'une extension pure et simple de la programmation dynamique, c'est-à-dire $O(n^N)$. En pratique cependant, les méthodes de cette section permettent d'aligner de 6 à 8 chaînes en un temps 'raisonnable'.

4.1.2.2.3 Une nouvelle approche plus orientée graphe

Signalons une autre approche, également exacte et NP-complète, de l'alignement d'un ensemble de chaînes qui est formalisée de manière différente, faisant plus exclusivement appel à la théorie des graphes (recherche d'un chemin mais aussi calcul de flots), et qui a été proposée par Kececioglu [Kececioglu, 1993]. Le problème d'un alignement multiple de chaînes est dans ce cas modélisé par celui d'une trace de poids maximal dans un graphe d'alignement. Un tel graphe est composé d'un ensemble N de nœuds représentant les symboles dans les chaînes, d'un ensemble A d'arêtes liant deux nœuds si les symboles correspondant à ceux-ci se situent sur des chaînes différentes (quelles que soient leurs positions) et sont identiques, et enfin d'un ordre $\prec$ sur les nœuds où $v \prec w$ pour $v, w \in N$ si et seulement si le symbole v vient immédiatement avant le symbole w dans une des chaînes. Un sous-graphe connexe dans un tel graphe représente une colonne de symboles identiques et donc une colonne dans un alignement possible des chaînes. Un ensemble de tels sous-graphes connexes forme alors un alignement valable de ces chaînes, appelé une trace, s'ils peuvent être ordonnés de façon à respecter l'ordre $\prec$ symbole par symbole. Pour vérifier cela, l'ordre $\prec$ est en fait étendu à un ordre $\prec^*$ sur les sous-graphes connexes où $c_1 \prec^* c_2$ pour c_1 et c_2 des sous-graphes connexes du graphe s'il existe au moins une paire de symboles v et w dans c_1 et c_2 respectivement tels que $v \prec w$. Une trace dans un tel graphe est dans ce cas défini comme étant un sous-ensemble T de A tel que la relation $\prec^*$ sur les composantes connexes de T est acyclique. Rechercher une trace de poids maximal correspond à rechercher un alignement optimal sur la base de scores SP et est effectuée par une technique d'exploration exhaustive avec bornes. En pratique, l'algorithme permet d'aligner jusqu'à 6 chaînes de longueur environ 250 symboles en quelques minutes, sa performance en moyenne est donc similaire aux méthodes de programmation dynamique plus classiques. Cette approche peut se révéler par contre plus intéressante que les précédentes si les symboles sont remplacés par des mots communs identiques ou similaires et si la relation $\prec^*$ devient une relation sur ces mots.

4.1.2.3 Algorithmes exacts utilisant un arbre phylogénétique

4.1.2.3.1 Application directe de la récurrence — problème de la complexité

Les algorithmes exacts utilisant un arbre phylogénétique qui ont été élaborés à ce jour supposent, ainsi que nous l'avons vu, que cet arbre soit connu à l'avance. Cet arbre est composé de nœuds externes correspondant aux feuilles et où sont placées les chaînes de l'ensemble de départ et de un ou plus de nœuds internes associés à des chaînes ancestrales initialement inconnues. Un arbre avec un seul nœud interne est appelé une étoile [Altschul and Lipman, 1989] (voir la section 3.4.1.5). Ce qui est recherché est un alignement optimal des chaînes de départ, c'est-à-dire un alignement correspondant à un nombre minimal de changements (substitutions, délétions et insertions) par rapport à toutes les chaînes, celles de départ plus les chaînes ancestrales.

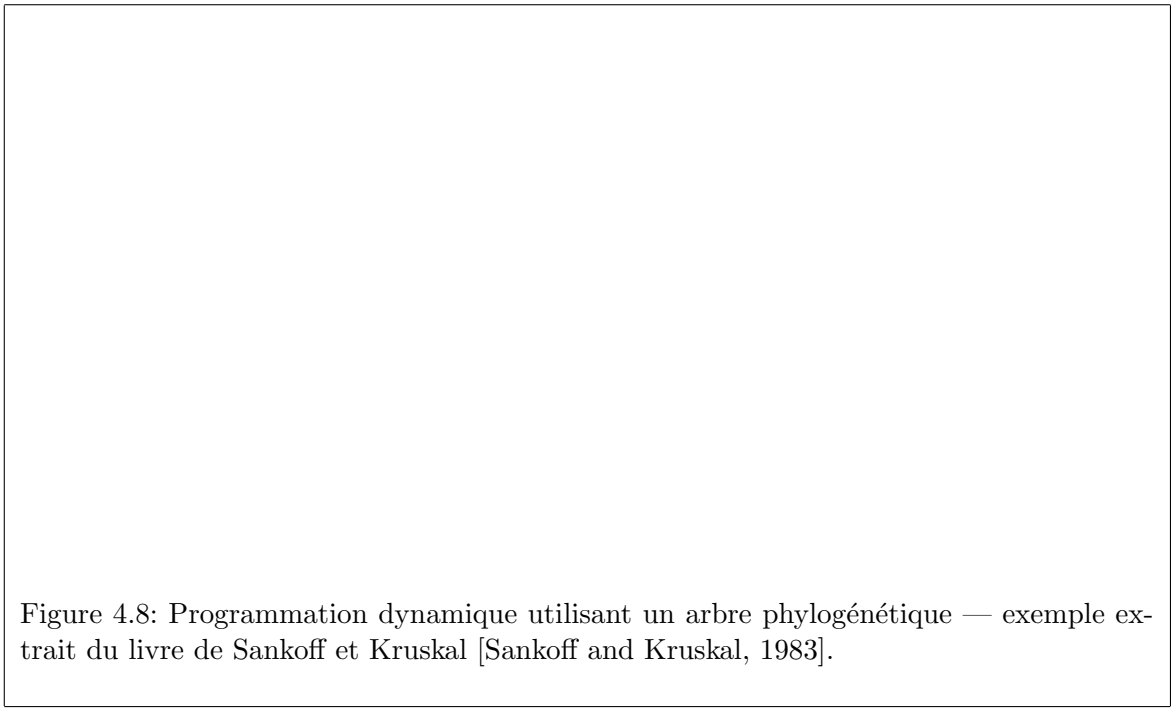


Figure 4.8: Programmation dynamique utilisant un arbre phylogénétique — exemple extrait du livre de Sankoff et Kruskal [Sankoff and Kruskal, 1983].

Le premier algorithme pour résoudre ce problème a été conçu par Sankoff [Sankoff, 1975]. Le processus de construction de l'alignement est inductif comme pour l'alignement de plusieurs chaînes utilisant des scores SP. Le score de l'alignement optimal de N chaînes jusqu'aux positions $i_1, ..., i_N$ est ainsi égal au minimum du score de l'alignement optimal obtenu un cran avant, c'est-à-dire, jusqu'aux positions i_j ou $i_j - 1$, plus une minimisation interne qui représente la plus petite distance d'édition pour, étant donné une certaine attribution de symboles des chaînes de départ ou de trous aux feuilles, toute attribution possible de symboles aux nœuds internes, y compris des trous (voir la page 87 ainsi que l'illustration de la figure 4.8).

Si $\mathcal{O}$ est un alignement optimal des N chaînes, on obtient ainsi à la fin

$$score(\mathcal{O}) = \sum_{a \in A} score(\mathcal{O}_a)$$

où A est l'ensemble des arêtes de l'arbre et $score(\mathcal{O}_a)$ est le score de la projection de $\mathcal{O}$ sur l'arête a.

L'algorithme de Sankoff est en $O((2n)^N M)$ où n et N sont comme avant la longueur moyenne et le nombre des chaînes respectivement et M est le nombre des chaînes ancestrales (correspondant au nombre de nœuds internes de l'arbre donné au départ).

4.1.2.3.2 Économies possibles

Comme dans le cas des scores SP, il est possible d'essayer d'économiser sur le nombre d'opérations à réaliser en estimant des bornes supérieures aux scores des projections d'un alignement $\mathcal{H}$ obtenu par une heuristique sur chacune des paires de chaînes de l'ensemble de départ [Altschul and Lipman, 1989]. Le score de la projection sur chaque paire (si, sj) est cependant donnée désormais uniquement par le coût de la projection du score de $\mathcal{H}$ sur chacune des arêtes

de l'arbre reliant si à sj. Pour obtenir une estimation de la borne supérieure au score d'une telle projection de $\mathcal{H}$, on va comme auparavant soustraire du score de $\mathcal{H}$ la somme des distances entre toutes les paires (sk, sl) autres que (si, sj) puisque chacune de ces distances représente une borne inférieure à la distance réelle, mais imposée par l'arbre, entre sk et sl. Il va y avoir cependant une importante différence avec le cas précédent. En effet, certains chemins entre paires quelconques de chaînes peuvent posséder des arêtes en commun et il faut s'assurer que le coût de celles-ci n'est retiré du score de $\mathcal{H}$ qu'une seule fois. À ce détail près, le calcul des bornes supérieures pour la projection de $\mathcal{H}$ sur chaque paire de chaînes de l'ensemble de départ permet, comme pour les scores SP, de diminuer le nombre de cellules dont il faut déterminer la valeur dans le tableau de programmation dynamique.

4.1.3 Les heuristiques

4.1.3.1 Deux catégories d'heuristiques

Tous les algorithmes de comparaison ou alignement multiple que nous venons de voir sont exponentiels dans le nombre de chaînes à analyser. Ainsi que nous l'avons fait observer, le problème dans sa plus grande généralité est NP-complet et les astuces permettant d'effectuer moins de calculs dépendent fortement de la nature des données à traiter.

Il s'est avéré nécessaire par conséquent d'élaborer des heuristiques pour comparer globalement un ensemble de chaînes dans le but de les aligner. L'appel à une heuristique introduit toujours la nécessité d'un choix. Celui-ci peut s'appuyer sur un critère purement algorithmique (ne pas explorer tout l'espace de recherche) ou purement biologique (ne pas prendre les chemins dont on peut supposer dès le départ qu'ils conduiront à des résultats qui ne sont pas intéressants). En général les deux critères sont mélangés et la division en deux catégories que nous présentons ici, bien que paraissant suivre la ligne esquissée plus haut, ne lui est pas tout à fait fidèle. Un choix effectué sur la base de critères purement algorithmiques peut ainsi être biologiquement tout à fait justifiable, et vice-versa, un choix réalisé pour des motifs biologiques peut conduire à une manière de réduire l'espace à explorer qui est clairement exprimable en termes mathématiques.

La première catégorie d'heuristiques que nous présentons concerne alors celles qui demeurent proches des algorithmes exacts dans le sens où elles adoptent les mêmes définitions de la ressemblance que celles-ci. Elles travaillent ainsi d'une part avec des scores SP ou des scores obtenus sur la base d'un arbre phylogénétique et, d'autre part, elles cherchent à s'approcher au mieux du résultat obtenu par les algorithmes d'optimisation d'une fonction de score. Parmi celles qui ont été proposées, deux sont particulièrement intéressantes dans la mesure où justement elles garantissent sinon l'optimalité du résultat obtenu, du moins une certaine 'qualité' à ce résultat où par qualité on entend, une distance maximale du résultat considéré optimal par rapport aux définitions de ressemblance adoptées. Les économies que toutes réussissent à réaliser résultent du fait que, soit elles réduisent le nombre de chaînes à comparer (c'est le cas de l'heuristique utilisant le concept d'étoile centrale), soit elles placent une contrainte sur le type d'arbre phylogénétique autorisé.

Les heuristiques de la seconde catégorie s'éloignent quant à elles de plus en plus des algorithmes exacts, cela de deux façons. La première concerne la mesure de similarité multiple utilisée. Le score d'un alignement de plus de deux chaînes représente ainsi rarement la somme des scores de tous les alignements deux à deux induits, ou la somme des scores des alignements deux à deux suivant un arbre. La seconde différence avec les méthodes exactes est liée à la première d'une certaine manière puisque n'ayant plus de fonction de score nettement établie, il n'est plus question d'optimiser globalement l'alignement multiple. La plupart de ces heuristiques

continue cependant d'optimiser le score de tous les alignements deux à deux. Ceux-ci peuvent concerner non seulement deux chaînes mais faire référence aussi à une chaîne et à un alignement ou à deux alignements préalablement obtenus. Il est important d'observer que les plus soigneusement conçues parmi ces heuristiques réussissent parfois à cerner certains phénomènes biologiques, particulièrement ceux liés à l'évolution, de manière souvent plus fine que les heuristiques de la première catégorie, ou même que les algorithmes exacts. Nous avons en effet fait remarquer (section 4.1.1.1) que les alignements ayant un score optimal ne correspondent pas toujours aux alignements qui sont 'corrects' biologiquement. Le motif en est fondamentalement que certaines définitions de ressemblance données au chapitre précédent, bien qu'établies sur la base de critères mathématiques clairs et précis, représentent parfois une simplification des mécanismes biologiques.

4.1.3.2 Les heuristiques proches des algorithmes exacts

4.1.3.2.1 Les heuristiques sans garantie de la qualité de l'alignement obtenu

D'une certaine façon, nous avons déjà présenté des heuristiques au calcul du score d'un alignement optimal dans la mesure où les algorithmes exacts qui essayent de réduire le nombre de cellules à calculer dans le tableau de programmation dynamique utilisent parfois des bornes supérieures 'inexactes'. C'est le cas par exemple de Altschul *et al.* et de Gupta *et al.* pour les alignements ayant pour base des scores SP.

Les deux heuristiques que nous présentons ici cependant ne s'appuient pas sur le calcul de bornes supérieures ou inférieures. Ce ne sont donc plus des techniques de 'branch-and-bound'. Par ailleurs, ces heuristiques travaillent avec des scores obtenus à partir d'un arbre et non plus des scores SP. Elles ont pour principe l'adoption d'un arbre phylogénétique qui permette de simplifier les calculs. Une première approche est due à Sankoff et Cedergren [Sankoff and Cedergren, 1983] et utilise la formule de récurrence donnée à la page 87 du chapitre précédent. Celle-ci comporte une double minimisation (dans le cas d'un arbre, on travaille avec des mesures de distances) et l'idée de Sankoff et Cedergren est de limiter le nombre de calculs à effectuer pour obtenir la minimisation interne, c'est-à-dire limiter le nombre de comparaisons à effectuer entre chaînes de départ et chaînes ancestrales. Il faut pour cela choisir comme arbre phylogénétique un arbre décomposable en un ensemble de sous-arbres qui ont chacun trois branches et qui se superposent, puis appliquer l'algorithme exact uniquement à chacun de ces sous-arbres pris dans un certain ordre. En termes pratiques, ce qui change dans la façon de calculer le min interne de la formule est que, au lieu de réaliser, pour toute affectation possible de symboles des chaînes de départ (ou de trous) aux feuilles de l'arbre, toutes les attributions possibles de symboles ou trous aux nœuds internes, un de ceux-ci, notons-le x, est choisi initialement au hasard parmi ceux qui sont liés à deux autres nœuds auxquels des symboles a et b ont déjà été attribués. Un symbole est alors affecté à x suivant la valeur de a et b, puis un second nœud interne est choisi vérifiant les mêmes conditions que le nœud x et ainsi de suite. La qualité de l'alignement obtenu à la fin dépend bien sûr fortement de l'ordre avec lequel les nœuds internes sont sélectionnés (dans le cas où plusieurs choix sont possibles) pour se voir affecter un symbole ou un trou. La complexité de l'algorithme de Sankoff et Cedergren est $O((2n)^3.N)$.

Une approche différente est fournie par Waterman et Perlwitz [Waterman and Perlwitz, 1984] et permet d'aligner N chaînes de longueur moyenne n en $O(N.n^2)$. L'arbre phylogénétique utilisé est dans ce cas un arbre binaire et un concept nouveau appelé une 'chaîne moyenne pondérée' est employé (ces chaînes sont situées aux nœuds internes de l'arbre). Une chaîne

moyenne pondérée de plusieurs chaînes définies sur un alphabet Σ est une chaîne $a = a_1 a_2 ... a_k$ où chaque a_i a la forme $a_i = (p_0, p_1, p_2, ..., p_{|\Sigma|})$, avec p_0 correspondant à la proportion de trous dans les chaînes alignées résumées par la chaîne a et relatifs à la position i de a, et p_j pour $j > 0$ représentant la fréquence du $j^{ème}$ élément de Σ dans cet alignement, toujours relativement à la position i de a. Par conséquent, on a $p_j \geq 0$ et $\sum_{j>0} p_j = 1$. La ressemblance de deux 'lettres' $a = (p_0, p_1, ..., p_{|\Sigma|})$ et $b = (q_0, q_1, ..., q_{|\Sigma|})$ est donnée par :

$$d(a, b) = (\sum_{i \geq 0} w_i. \mid p_i - q_i \mid^{\alpha})^{\frac{1}{\alpha}}$$

où les w_i sont des facteurs de pondération et $\alpha \geq 1$ est une constante. Il a été montré que d représente en fait une métrique. La méthode commence donc par construire la 'chaîne moyenne' de deux des chaînes de départ placées aux feuilles de l'arbre binaire et qui sont les fils d'un même nœud de cet arbre. Puis le processus de construction se poursuit en suivant les relations de dépendances de l'arbre jusqu'à la racine pour laquelle une chaîne moyenne pondérée finale est dérivée. L'alignement des chaînes de départ est obtenu alors par un alignement de celles-ci avec la chaîne moyenne finale. La complexité en temps de l'algorithme de Waterman et Perlwitz est $O(N.n^2)$ et celle en espace $O(n^2 + n.N)$.

4.1.3.2.2 Les heuristiques garantissant une certaine qualité à l'alignement obtenu

Les deux heuristiques que nous présentons maintenant ont été élaborées par Gusfield [Gusfield, 1993]. Comme pour les heuristiques de la section précédente, celles-ci ont pour principe limiter le nombre de chaînes à comparer, et donc le nombre de calculs à effectuer. Elles diffèrent des méthodes antérieures d'une manière importante cependant dans la mesure où elles sont capables également de fournir une borne supérieure à l'erreur qui est commise dans l'alignement obtenu, c'est-à-dire en fait dans le score de l'alignement produit par rapport à l'alignement optimal. Afin de pouvoir obtenir une telle borne, la mesure de ressemblance utilisée doit vérifier l'inégalité triangulaire. Pour faciliter les explications, nous considérons ici (dans les exemples en particulier) que cette mesure représente une simple distance d'édition bien que d'autres métriques ou ultramétriques puissent être utilisées (notons que la borne est indépendante de la mesure choisie).

Par rapport à un ensemble de départ comportant N chaînes, Gusfield garantit alors que :

$$score(\mathcal{A}) \leq (2 - \frac{2}{N}).score(\mathcal{O}) \leq 2.score(\mathcal{O})$$

où $\mathcal{O}$ est un alignement optimal des N chaînes et $\mathcal{A}$ l'alignement multiple réalisé par l'algorithme de Gusfield. Cette borne vaut pour les deux heuristiques qu'il présente bien que la première fasse appel à des scores SP et la seconde à des scores obtenus à partir d'un arbre.

La méthode s'appuyant sur des scores SP utilise le concept d'étoile centrale ('Center Star') que nous avons introduit dans le chapitre 3, section 3.4.1.5. Elle consiste à choisir à tour de rôle chacune des N chaînes de départ comme chaîne de référence, notons cette chaîne sc, et à considérer la somme

$$\sum_{\substack{i \in [1..N] \\ i \neq c}} distance(sc, si)$$

des distances de toutes les chaînes à sc. L'alignement des chaînes $s1, ..., sN$ fourni par l'algorithme est alors celui qui minimize cette somme et l'étoile centrale est une chaîne pour laquelle ce minimum est atteint (il peut y avoir plus d'une chaîne dans cette situation). Un exemple a été donné dans la figure 3.22.

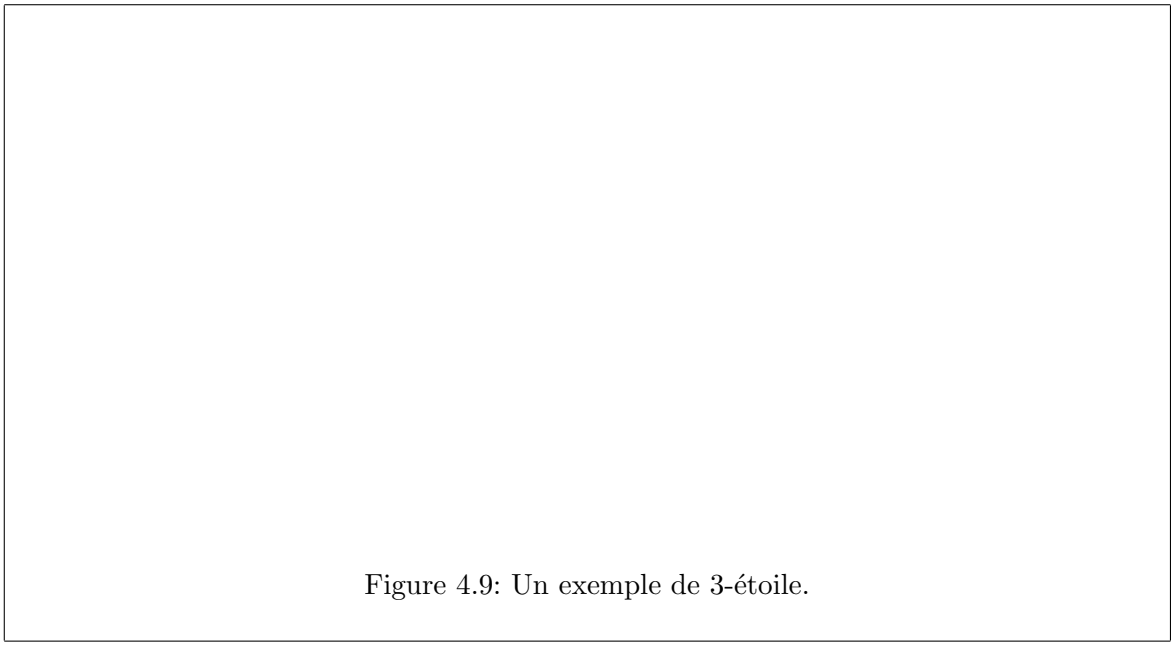


Figure 4.9: Un exemple de 3-étoile.

Comme nous l'avions fait observer alors, cette mesure de ressemblance peut être vue comme une approximation des scores SP ou comme une mesure intéressante et importante en soi. Gusfield lui-même la présente comme une approximation et c'est pourquoi nous avons placé son algorithme parmi les heuristiques. Nous argumenterons dans le chapitre 5 qu'une telle mesure de score multiple peut être plus valable dans certains cas, en particulier pour les comparaisons locales, que les scores SP.

Comme l'algorithme de Gusfield s'appuie sur un calcul de toutes les distances deux à deux, sa complexité en temps est en $O(n^2.N^2)$. Si uniquement $p < N$ chaînes sont choisies au hasard comme candidates possibles à devenir des étoiles centrales, l'erreur commise peut être plus grande mais Gusfield garantit néanmoins que l'on aura :

$$score(\mathcal{A}) \leq ((2 - \frac{2}{N}) + (\frac{N}{p} - \frac{1}{p})).score(\mathcal{O}).$$

Enfin, en travaillant non avec des alignements deux à deux mais avec des alignements L à L (c'est-à-dire, en construisant une L-étoile — voir figure 4.9 pour $L = 3$) Pevzner [Pevzner, 1992a] [Pevzner, 1992b] et Bafna [Bafna *et al.*, 1994] montrent que la borne pour l'erreur devient :

$$score(\mathcal{A}) \leq (2 - \frac{L}{N}).score(\mathcal{O}).$$

La complexité de l'algorithme est dans ce cas $O(N^3.n^L)$.

La seconde heuristique introduite par Gusfield cherche à construire un alignement des chaînes de départ en utilisant cette fois un arbre. Contrairement à ce qui se passe dans l'approche de Sankoff ou celle de Waterman, cet arbre ne comporte pas d'ancêtres. Les chaînes de l'ensemble se situent ainsi non seulement aux feuilles mais aussi aux nœuds internes de l'arbre. Aucun nœud interne correspond donc à une chaîne qui serait inconnue initialement. La méthode consiste alors à partir d'un graphe complet G à N nœuds dont chacun représente une chaîne de l'ensemble de départ et où chaque arête a pour poids le score d'un alignement optimal

Figure 4.10: Un exemple de l'heuristique de Gusfield utilisant un arbre.

des deux chaînes liées par l'arête, et à chercher un arbre couvrant de score minimal de G (en anglais 'minimal spanning tree'). Il s'agit d'un arbre qui contient tous les nœuds de G et dont la somme des arêtes est la plus petite possible (voir la figure 4.10). Des algorithmes classiques en théorie des graphes (dus à Kruskal et Prim notamment [Kruskal, 1956] [Prim, 1957]) permettent de déterminer un tel arbre en un temps $O(N^2)$ (pire cas pour l'algorithme de Prim). L'alignement final des N chaînes est celui qui est consistant avec cet arbre. C'est donc celui dont le score de la projection sur chaque paire de chaînes (si, sj) est égal à la somme du poids des arêtes liant si et sj dans l'arbre. Comme toutes les distances entre paires de chaînes doivent être calculées, la complexité de l'algorithme est $O(n^2.N^2 + N^2)$, c'est-à-dire $O(n^2.N^2)$.

4.1.3.3 Les heuristiques par alignements progressifs

4.1.3.3.1 Introduction

Les heuristiques de la seconde catégorie (voir la section 4.1.3.1) possèdent un premier point en commun dans la mesure où toutes effectuent un alignement progressif des chaînes d'un ensemble de départ. Elles commencent donc par aligner une chaîne avec une autre, puis une chaîne avec un alignement obtenu au préalable, parfois un alignement avec un autre alignement. Le score d'un alignement de plus de deux chaînes ne représente donc que rarement la somme des scores de tous les alignements deux à deux induits. Ces heuristiques optimisent presque toujours les scores de tous les alignements deux-à-deux mais il n'y a plus d'optimisation globale de l'alignement multiple.

Ainsi que le signale Gusfield [Gusfield, 1995], ces méthodes progressives déterminent une suite de fusions de sous-ensembles disjoints des chaînes de départ qui peut être résumée par un arbre. Si les divers critères utilisés pour fusionner les sous-ensembles de chaînes sont supposés refléter l'histoire évolutive de celles-ci (par exemple, s'ils s'appuient sur l'hypothèse que des chaînes plus similaires entre elles sont également plus proches évolutivement), alors l'arbre obtenu représente non seulement une description du processus de construction de l'alignement final mais est aussi un arbre phylogénétique déduit des séquences fournies en entrée de l'algo-

rithme. Les méthodes progressives d'alignement peuvent ainsi jouer un rôle important aussi bien dans la classification des macromolécules et l'identification de structures ou fonctions communes, que dans la déduction de phylogénies. Toutes ces approches sont donc bien sûr motivées par la recherche d'une plus grande efficacité algorithmique, mais certaines représentent aussi une réflexion parfois profonde sur la ressemblance de macromolécules et qui n'est pas moins valable du fait qu'elle est moins clairement définissable (une question ouverte — et que pose en tout cas Higgins [Thompson *et al.*, 1994a] — est alors de savoir si une telle réflexion pourrait être formalisée).

Nous pouvons répartir ces heuristiques par alignements progressifs en deux grandes classes qui sont :

1. celles qui effectuent de tels alignements en suivant un certain ordre, en général séquentiel, établi sur les chaînes de départ et qui alignent toujours deux chaînes ou une chaîne avec un alignement réalisé au préalable;

2. celles qui essayent de regrouper les chaînes suivant leur degré de ressemblance avant de les aligner (qui essayent en fait, implicitement ou explicitement, d'estimer un arbre phylogénétique pendant le déroulement de l'algorithme) et qui sont ainsi amenées à effectuer des alignements d'alignements.

Les secondes diffèrent des premières essentiellement parce que les arbres qu'elles construisent sont plus complexes. Il est important d'observer que toutes ces approches, particulièrement celles de la seconde classe, sont en fait des heuristiques à mettre en rapport avec les algorithmes exacts qui chercheraient à obtenir un alignement multiple sur la base d'un arbre qui ne serait pas connu à l'avance.

Notons qu'un second point en commun présenté par ces heuristiques vient du fait qu'elles ont besoin de pré-calculer les scores de tous les alignements deux à deux des chaînes de départ, ce qui explique que le terme dominant dans leur complexité soit le plus souvent en $n^2.N^2$. Les constantes peuvent bien sûr varier considérablement d'une méthode à l'autre, elles ne sont pas discutées ici.

4.1.3.3.2 Alignements progressifs suivant un ordre séquentiel

La façon sans doute la plus naturelle de procéder afin d'aligner N chaînes suivant un ordre séquentiel consiste à attribuer à chaque paire (si, sj) un score représentant une mesure de similarité ou de dissimilarité (ou distance) entre si et sj et à classer les paires d'après ces scores [Barton and Sternberg, 1987a] [Barton and Sternberg, 1987b] [Barton, 1990] [Barton and Sternberg, 1990] [Subbiah and Harrison, 1989]. Les deux chaînes les plus proches (les plus 'semblables') sont alors alignées en premier, notons-les $s1$ et $s2$, puis on aligne avec l'alignement de $s1$ et $s2$ la chaîne parmi celles qui demeurent qui est la plus proche soit de $s1$, soit de $s2$ et ainsi de suite. À la fin de chaque étape, lorsqu'un alignement de p parmi les N chaînes de départ a été obtenu, une série précise de raffinements successifs peut être appliquée à cet alignement.

Ainsi, si $3 \leq p \leq N$ chaînes ont été alignées dans un certain ordre ($s1$ avec $s2$, puis $s3$ avec l'alignement de $s1$ et $s2$, et ainsi de suite jusqu'à sp avec l'alignement de $s1, ..., s(p-1)$), cet alignement est 'gelé' en relation à tous les sous-ensembles possibles de q parmi les p chaînes pour q allant de 2 à $p-1$, et les $p-q$ chaînes restantes réalignées par rapport à l'alignement 'gelé' des q. Le coût excessif d'une telle opération (qui demande $O(N!)$) comparaisons deux à deux entre chaînes ou entre alignements) fait qu'en pratique quelques-uns seulement parmi tous les réalignements possibles sont effectués, ou bien alors le processus de raffinements est

arrêté avant la fin. Subbiah opte pour la seconde approche puisqu'il fixe à l'avance le nombre maximal de fois où le processus peut être répété. Dans le cas de Barton par contre, la limitation est réalisée en fixant q égal à $p-1$, il n'y a donc jamais qu'une chaîne qui est 'libérée' et réalignée avec les $p-1$ autres.

Notons que dans les deux cas, l'ordre parmi les paires de chaînes peut être établi sur la base non de leurs scores de ressemblance mais plutôt de la signification statistique de ces scores où cette signification est calculée par rapport aux chaînes de l'ensemble, les symboles de chacune étant ré-ordonnés de manière aléatoire.

Enfin, Bains [Bains, 1986] [Bains, 1989] réalise des alignements progressifs suivant un ordre non complètement séquentiel. Dans son cas, une chaîne, appelons-la $s0$, est initialement privilégiée parmi toutes les autres et c'est l'ordre suivant lequel ces autres 'ressemblent plus' à $s0$ qui détermine laquelle va être sélectionnée pour être alignée avec $s0$, ou avec un alignement de $s0$ et des chaînes les plus proches de $s0$ qui auront été choisies au préalable. Une fois un premier alignement des N chaînes obtenu, une chaîne consensus est dérivée de cet alignement. Si elle est différente de la chaîne $s0$ de la première étape, elle prend sa place et tout le processus est répété jusqu'à convergence. Celle-ci peut bien sûr ne jamais se produire si, par exemple, les chaînes à aligner sont trop dissemblables, ou si le choix initial de $s0$ conduit à un processus qui boucle sur lui-même.

4.1.3.3.3 Alignements progressifs regroupant les chaînes ou estimant un arbre

Parmi les méthodes qui éventuellement regroupent certaines des chaînes de départ [Corpet, 1988] [Feng and Doolittle, 1987] [Feng and Doolittle, 1990] [Higgins and Sharp, 1988] [Higgins and Sharp,1989] [Higgins *et al.*, 1992] [Hogeweg and Hesper, 1984] [Smith and Smith, 1990] [Smith and Smith, 1992] [Taylor, 1987] [Taylor, 1990] [Thompson *et al.*, 1994a] [Thompson *et al.*, 1994b] [Wong *et al.*, 1993] la différence porte d'abord sur la façon d'effectuer ces regroupements. Comme pour les approches décrites dans la section précédente, initialement les scores de tous les alignements deux à deux des chaînes de l'ensemble sont calculés. Signalons que dans le cas de Higgins cependant, ce calcul est approché et non exact car il veut pouvoir traiter un très grand nombre de chaînes simultanément. Les chaînes peuvent alors être regroupées dès le début ou bien les groupes se former au fur et à mesure que l'alignement de l'ensemble se poursuit. Cela est le cas de Corpet en particulier qui commence par aligner les deux chaînes les plus proches, puis la matrice de tous les scores deux à deux est en partie recalculée avec la ligne correspondant à la chaîne $s2$ supprimée et celle correspondant à la chaîne $s1$ substituée par la moyenne des scores des lignes relatives à $s1$ et $s2$. Sont alignées ensuite les deux chaînes, ou la chaîne et un alignement préalable ou encore les deux alignements qui sont les plus proches d'après cette nouvelle matrice. Feng, Higgins, Hogeweg, Smith, Taylor et Wong de leur côté suivent la première approche, c'est-à-dire ils peuvent être amenés à effectuer les regroupements dès l'étape initiale et font en fait appel explicitement à des algorithmes de construction d'arbres phylogénétiques [Blanken *et al.*, 1982] [Klotz *et al.*, 1979] [Klotz and Blanken, 1981] [Saitou and Nei, 1987] [Sneath and Sokal, 1973] ou de regroupements hiérarchiques basés sur des graphes aléatoires [Wong *et al.*, 1993]. L'algorithme de Smith *et al.* produit en outre à la fin une chaîne consensuelle qui est toutefois différente de la chaîne moyenne de Waterman (voir la section 4.1.3.2.1). Dans leur cas, cette chaîne est définie non plus sur l'alphabet des monomères mais sur une couverture qui représente une partition de cet alphabet. La chaîne est ainsi un motif dégénéré (joker $\{\Sigma\}$ non autorisé) dans le sens de l'exemple de la figure 3.24 du chapitre 3.

Parmi tous ces algorithmes, celui de Higgins composant la série des CLUSTALs (appelé CLUSTAL W dans sa dernière version) est de loin le plus utilisé. La raison de ce succès est

qu'il incorpore une richesse d'information dans son processus d'alignement qui n'est sans doute égalé par aucune autre méthode travaillant uniquement avec les séquences. Ainsi, bien que CLUSTAL W suive les mêmes principes que les méthodes antérieures (estimation d'un arbre, regroupement des chaînes selon cet arbre, alignement des chaînes intra puis inter-groupes), un soin très grand est accordé à tous les détails de la construction de l'arbre et au choix des valeurs à attribuer aux divers paramètres pouvant avoir une influence sur la qualité de l'alignement final, en particulier le poids à affecter aux trous. En ce qui concerne l'arbre par exemple, ce n'est pas seulement la position des chaînes les unes par rapport aux autres qu'Higgins *et al.* essayent de déterminer ainsi que le font les autres (c'est-à-dire la forme de l'arbre) mais aussi la position absolue de ces chaînes (recherche de l'emplacement de la racine de l'arbre) et le degré de leur dérive relative (affectation d'un poids aux branches de l'arbre suivant les distances des chaînes par rapport à l'ancêtre commun présumé placé à la racine). Ce dernier point permet à la fois de tenir compte de la présence éventuelle d'un biais dans l'ensemble à aligner (si plusieurs des chaînes sont très proches les unes des autres par exemple) et d'utiliser des matrices de substitutions appropriées suivant la distance séparant les chaînes. Le poids à attribuer aux trous quant à lui varie suivant que le trou vient d'être ouvert ou qu'il est simplement allongé, il dépend également de la distance séparant les chaînes (le poids est plus élevé pour les chaînes plus proches), de la longueur de ces chaînes (ce qui affecte le coût relatif à l'ouverture d'un trou), de la position du trou (le poids est plus élevé pour un trou introduit dans un alignement à un endroit où jusqu'alors il n'y en avait pas, il est plus faible dans les régions hydrophiles des protéines qui correspondent le plus souvent aux boucles etc) et enfin de l'environnement du trou (les monomères situés juste avant et après le trou). Comme le dit clairement Higgins, en particulier dans [Thompson *et al.*, 1994a], ces détails de construction de l'arbre et choix des valeurs à affecter aux paramètres ne sont guidés que par l'observation et l'expérience et demeurent donc de nature empirique pour l'instant.

4.1.3.3.4 Avantages et inconvénients

Indépendamment des informations supplémentaires qui peuvent être incorporées dans un alignement par ces méthodes d'alignements progressifs du fait qu'elles ne cherchent plus à optimiser une fonction particulière, une idée dominante qui est commune à toutes et sur laquelle nous voulons insister parce qu'elle est importante, est celle de comparer d'abord les objets qui se ressemblent le plus, et puis d'une certaine façon d"apprendre' quelque chose de cette première comparaison avant de se lancer à comparer les autres objets d'un ensemble. Cette idée est sans doute illustrée le mieux par la phrase de Feng [Feng and Doolittle, 1987] : " *once a gap — always a gap*" qui signifie qu'une fois qu'un trou a été introduit dans l'alignement de p des N chaînes de départ, celui-ci devrait être conservé dans l'alignement final si les p chaînes alignées en premier représentent celles qui sont les plus proches les unes des autres. Cette phrase de Feng qui ne s'applique qu'aux trous peut en fait être généralisée à d'autres facteurs jouant pour la qualité d'un alignement et ce qu'elle exprime de la façon la plus large est que la comparaison d'un ensemble d'objets devrait commencer par essayer de distinguer les parties communes qui sont les 'plus similaires'. Cette idée n'est pas incontestable. On peut argumenter en effet que l'identification des blocs fortement similaires (comportant moins d'erreurs par exemple) qui ne seraient présents que dans un certain pourcentage des objets que l'on veut comparer pourrait empêcher de repérer des blocs plus faiblement similaires (c'est-à-dire, pr'esentant plus d'erreurs) mais communs à pourcentage plus grand de ces objets. Cependant l'idée de comparer ce qui est le plus similaire d'abord est une idée puissante et s'applique aussi bien au cas global (comparer d'abord les chaînes qui se ressemblent le plus) qu'au cas local (repérer les mots communs

correspondant à ce qui est le mieux conservé parmi un ensemble de ces chaînes). Dans le cas local cependant, localiser les similarités les plus fortes (ou les plus longues) d'abord n'empêche que temporairement de trouver les plus faibles (ou les plus courtes), et peut en fait être utilisée pour faciliter cette seconde recherche.

Pour en finir avec l'approche globale et progressive, observons que, outre son manque de formalisme, celle-ci présente trois principaux inconvénients qui sont :

1. le fait que l'alignement final dépend de l'ordre dans lequel les alignements intermédiaires sont effectués — ceci est particulièrement vrai pour les approches séquentielles mais s'applique aussi aux méthodes qui établissent un arbre phylogénétique;

2. bien qu'aucune de ces approches ne cherche vraiment à optimiser un score ou un alignement, toutes peuvent conduire à des situations de 'blocage' où l'algorithme produit un alignement de médiocre qualité et ne réussit plus à progresser vers un meilleur alignement;

3. la qualité des alignements réalisés dépend fortement du choix effectué sur certains paramètres (comme les scores de substitution, les poids à affecter aux trous etc), la valeur à attribuer à ces paramètres étant particulièrement difficile à évaluer lorsqu'il s'agit non plus d'aligner deux chaînes, mais d'aligner une chaîne avec un alignement antérieur, ou d'aligner 2 alignements précédemment obtenus.

Observons que ce dernier problème concernant la difficulté de choisir les poids à affecter à divers paramètres, bien qu'il soit plus aigu ici, est vrai aussi dans le cas des algorithmes exacts (voir le chapitre 3). Nous pouvons ainsi définir de manière mathématiquement très précise ce que signifie l'alignement optimal de plusieurs chaînes, mais la définition d'une fonction de score multiple repose en dernière instance sur le choix des scores deux à deux. Un changement dans ces scores change l'alignement, même si celui-ci demeure optimal par rapport au choix qui a été fait. Plusieurs des définitions de ressemblance que nous avons données au chapitre 3 ont pour base ces scores, qui ne s'appuient en fin de compte sur aucune notion qui paraisse biologiquement tout à fait fiable.

4.2 Approche locale

4.2.1 Introduction

Tout ce que nous avons présenté dans la section 4.1 concernait les méthodes directes de comparaison globale de chaînes. Or notre intérêt porte surtout sur les comparaisons locales. Les approches que nous venons de décrire sont importantes cependant car en fait la plupart du temps les méthodes locales ne sont que des adaptations de celles utilisées pour comparer des chaînes dans leur totalité. D'une manière très grossière, nous pouvons dire que la différence porte souvent simplement sur le moment où s'arrête une comparaison. Le principe de base est ainsi le même pour un grand nombre d'approches locales bien que les variations puissent être considérables.

Lorsque nous avons présenté les méthodes globales, nous avons aussi commencé par montrer les algorithmes exacts et ce n'est qu'ensuite que nous avons parlé des heuristiques. La raison en est qu'historiquement c'est à peu près ainsi que les choses ont été développées. Les chercheurs ont d'abord été très 'ambitieux', ensuite ils ont du revoir leur ambition à la baisse. Les méthodes locales que nous allons décrire maintenant sont venues après les globales, ce n'est en effet que plus tard que l'on s'est rendu compte que la recherche des parties communes, et de ces parties seulement, pouvait être à la fois intéressante en soi et souvent aussi plus appropriée que la

recherche d'une ressemblance globale. Or lorsque l'on en est arrivé là quelques leçons avaient déjà été apprises et ce sont ainsi les heuristiques qui ont été élaborées en premier. C'est donc celles-ci que nous présentons d'abord.

Avant cela, il convient d'observer que même lorsque les chercheurs sont d'accord pour dire que, dans certaines situations, ce sont les ressemblances locales qui sont les plus, ou parfois les seules, importantes à identifier, tous ne sont pas d'accord sur la meilleure façon de procéder pour repérer ces ressemblances. Ainsi Posfai [Posfai *et al.*, 1989] en particulier considère que la meilleure manière de localiser des traits communs à un ensemble de chaînes est de les comparer globalement d'abord, puis d'extraire de ces comparaisons (i.e. alignements) les 'morceaux' les plus caractéristiques. Cette opinion est fortement contestée par d'autres ainsi que le signale Gusfield [Gusfield, 1995] mais il est important de la mentionner. La méthode de Posfai est également celle employée par Barton [Barton, 1990] [Barton and Sternberg, 1990].

Comme pour l'approche globale, les heuristiques qui recherchent les similarités locales à un ensemble de chaînes peuvent être de divers types. Parmi ceux-ci, on distingue deux grandes classes : les heuristiques qui cherchent à optimiser une fonction de score et qui utilisent des formes modifiées de programmation dynamique pour le faire, et les méthodes d'optimisation stochastique. En ce qui concerne la première approche, nous allons devoir revoir un peu les techniques de programmation dynamique mais appliquées au cas local cette fois. Nous allons donc présenter les méthodes exactes d'abord mais celles-ci ne sont jamais utilisées en pratique et nous n'en parlons que parce que les méthodes approchées s'en servent.

4.2.2 Les heuristiques

4.2.2.1 Méthodes d'optimisation d'une fonction de score

4.2.2.1.1 Une parenthèse sur les méthodes exactes : la programmation dynamique appliquée au cas local

Dans la section 4.1.1.1, nous avons vu comment, étant donné une fonction de score, rechercher le score d'un alignement optimal global de deux chaînes $s1 = a_1...a_n$ et $s2 = b_1...b_m$. Si nous voulons déterminer maintenant le score d'un alignement optimal local des chaînes $s1$ et $s2$ au sens de la définition 3.4.7 présentée dans le chapitre 3, il faut commencer par rappeler que nous devons supposer que les scores avec lesquels nous travaillons mesurent une similarité entre monomères. De plus, il est impératif qu'au moins un de ces scores soit négatif. Le score optimal local est alors le plus élevé observé parmi tous les alignements possibles de toutes les paires de mots possibles dans les deux chaînes [Smith and Waterman, 1981]. La formule d'induction sur les valeurs du tableau est dans ce cas donnée par :

$$\begin{gathered}\mathcal{D}(i,j) = max\{0, \mathcal{D}(i-1,j-1) + score\begin{pmatrix} a_i \\ b_j \end{pmatrix}, \mathcal{D}(i-1,j) + w_\lambda, \mathcal{D}(i,j-1) + w_\lambda\} \\ \mathcal{D}(i,0) = 0 \\ \mathcal{D}(0,j) = 0\end{gathered}$$

pour $1 \leq i \leq n$ et $1 \leq j \leq m$. Le score recherché est alors égal à $max_{i,j}\mathcal{D}(i,j)$. Par rapport à la formule dans le cas global, le changement porte sur les valeurs frontières du tableau, ainsi que sur le fait que $D(i,j)$ est réinitialisé à zéro lorsque les trois autres termes de la formule sont négatifs. Une paire de mots (il peut y en avoir plus d'une) pour laquelle ce maximum est atteint est retrouvée en suivant le tableau des pointeurs qui aura été construit en même temps que le tableau des scores, depuis la cellule correspondant au score maximal jusqu'à la première cellule rencontrée ayant un score nul (voir la figure 4.11).

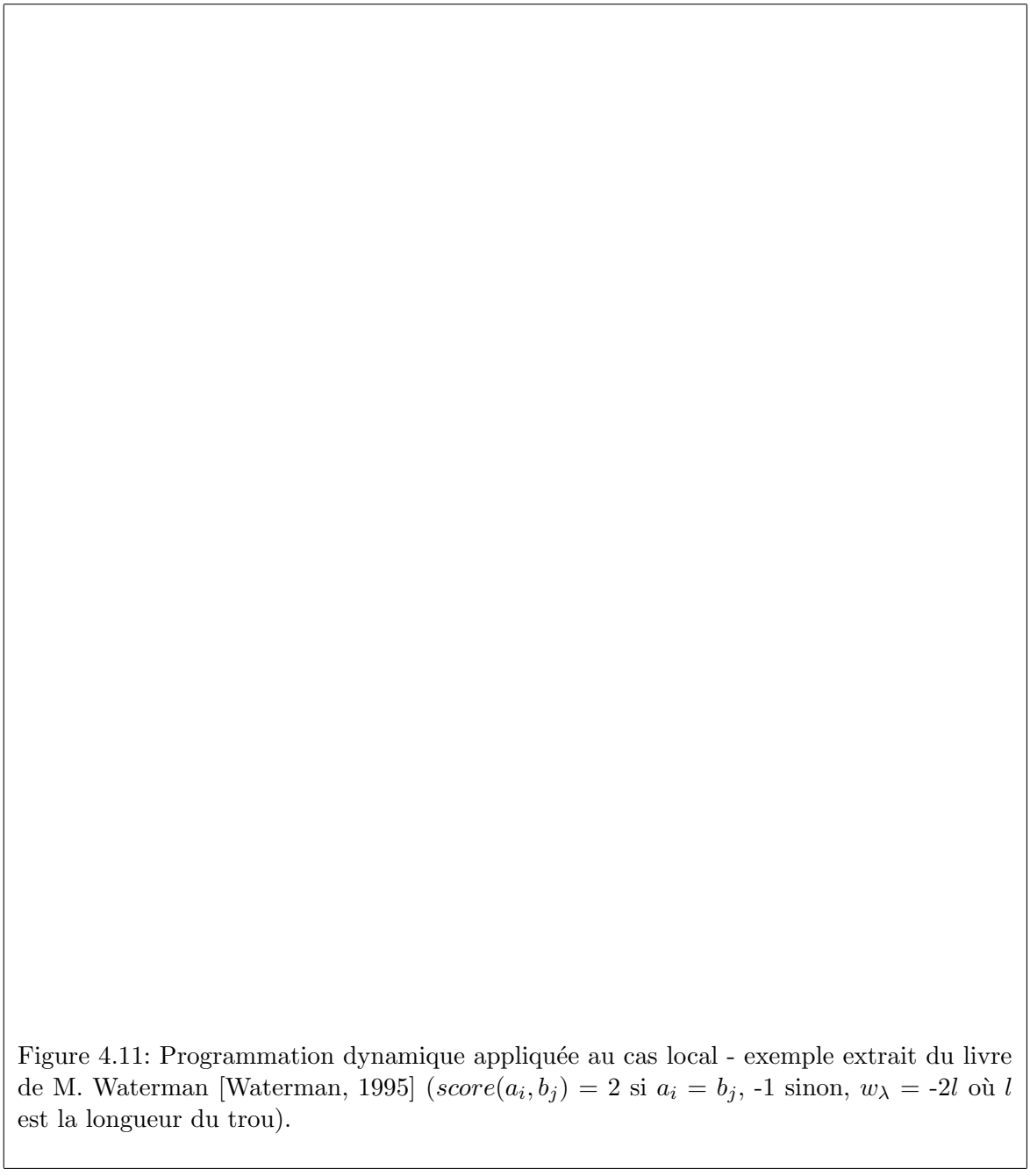


Figure 4.11: Programmation dynamique appliquée au cas local - exemple extrait du livre de M. Waterman [Waterman, 1995] ($score(a_i, b_j) = 2$ si $a_i = b_j$, -1 sinon, w_λ = -2l où l est la longueur du trou).

Il n'est pas tout à fait intuitif de comprendre pourquoi la formule de récurrence ci-dessus produit effectivement le résultat souhaité (le score d'une paire de mots la plus similaire parmi toutes les paires possibles, ainsi qu'une paire elle-même) aussi nous en fournissons une preuve.

Preuve que l'algorithme de Smith et Waterman est correct Prouvons d'abord que $max_{i,j}\mathcal{D}(i,j)$ est bien le score recherché.

Il suffit pour cela de prouver que $\mathcal{D}(i,j)$ représente pour $1 \leq i \leq n$ et $1 \leq j \leq m$ le meilleur score pour l'alignement optimal de toute paire de mots de $s1$ et $s2$ se terminant en i et j respectivement, c'est-à-dire, il suffit de prouver que :

$$\mathcal{D}(i,j) = max\ \{score(a_k...a_i, b_l...b_j) \mid 1 \leq k \leq i,\ 1 \leq l \leq j\}$$

où $score(a_k...a_i, b_l...b_j)$ est le score d'un alignement optimal global de $a_k...a_i$ et $b_l...b_j$.

Supposons par absurde que cela ne soit pas le cas. Il existe donc au moins une paire de mots $a_k...a_i$ et $b_l...b_j$ se terminant en i et j dans $s1$ et $s2$ respectivement avec $1 \leq k \leq i$ et $1 \leq l \leq j$ telle que $score(a_k...a_i, b_l...b_j)$ est strictement supérieur à $\mathcal{D}(i,j)$. Soit $\mathcal{D}'$ le tableau des scores/pointeurs de l'alignement global de $a_k...a_i$ et $b_l...b_j$. Quel que soit un chemin optimal dans $\mathcal{D}'$, ce chemin passe nécessairement par une cellule, notons-la (p,q) avec $k \leq p \leq i$ et $l \leq q \leq j$, telle que $score(a_k...a_p, b_l...b_q) < 0$ car si cela n'était pas le cas, nous aurions $\mathcal{D}'(i-k, j-l)$ $= \mathcal{D}(i,j)$ ce qui contredirait l'hypothèse que $\mathcal{D}'(i-k, j-l) = score(a_k...a_i, b_l...b_j) > \mathcal{D}(i,j)$. Mais alors $score(a_{p+1}...a_i, b_{q+1}...b_j)$ serait strictement supérieur au score de $a_k...a_i$ et $b_l...b_j$ puisque $score(a_k...a_i, b_l...b_j) = score(a_k...a_p, b_l...b_q) + score(a_{p+1}...a_i, b_{q+1}...b_j)$ ce qui à son tour contredirait l'hypothèse que $(a_k...a_i, b_l...b_j)$ est une paire de mots se terminant en i et j dans $s1$ et $s2$ respectivement ayant une valeur maximale dans le tableau $\mathcal{D}$.

Il faut maintenant prouver que la procédure permettant de retrouver une paire de mots pour laquelle ce maximum est atteint est correcte, c'est-à-dire qu'elle fournit bien une paire de mots ayant un score maximal parmi toutes les paires de mots possibles de $s1$ et $s2$. La preuve de cela est immédiate. En effet, soit $(a_k...a_i, b_l...b_j)$ une paire de mots retrouvée par la procédure. Il est clair que $score(a_k...a_i, b_l...b_j) = \mathcal{D}(i,j)$ puisque la liste des pointeurs est parcourue jusqu'à ce que soit rencontrée la première cellule du tableau contenant un zéro et que, par définition de la récurrence, le score de toute paire de mots se terminant en $k-1$ et $l-1$ dans $s1$ et $s2$ respectivement est négatif. Comme $\mathcal{D}(i,j)$ est le score optimal local maximal, alors $(a_k...a_i, b_l...b_j)$ est une paire de mots recherchée.

□

Les diverses économies qui peuvent être réalisées dans le calcul du score d'un alignement optimal de deux chaînes ne s'appliquent pas au cas des alignements locaux. Toutes les cellules de la matrice doivent donc être calculées, et l'algorithme de programmation dynamique est dans tous les cas en $O(n.m)$.

La détermination non pas uniquement du score d'un meilleur alignement, mais celle d'un alignement optimal lui-même possède la même complexité et est, comme nous venons de le voir, réalisée à l'aide d'un tableau de pointeurs construit en même temps que le tableau des scores. L'obtention par contre de tous les alignements optimaux locaux peut impliquer un recalcul des valeurs de la matrice (voir la figure 4.11 [Waterman and Eggert, 1987]). Il en est de même lorsque sont recherchés les c meilleures similarités locales, et dans ce cas la complexité est en $O(n.m.c)$ [Blum, 1990]. Toutefois ce recalcul n'est pas nécessaire lorsque seuls les c meilleurs scores nous intéressent (pas les alignements) et que le poids attribué à un trou est une fonction linéaire de sa longueur [Barton, 1993].

Enfin, en ce qui concerne la complexité en espace, elle est toujours en $O(n.m)$ lorsque les alignements sont recherchés, et en $O(min(m,n))$ lorsque l'on désire connaître uniquement le score optimal local [Barton, 1993] et que, à nouveau, le poids attribué à un trou est une fonction linéaire de sa longueur.

De même que le multiple global, le multiple local peut être vu comme une extension pure et simple de la programmation dynamique à N dimensions. *A priori*, rien n'empêche de considérer que les scores utilisés dans ce cas puissent être des scores calculés sur la base d'un arbre phylogénétique aussi bien que des scores SP. Les heuristiques qui s'appuient sur la programmation dynamique pour faire du multiple local utilisent cependant toujours des scores SP, et à notre connaissance aucune tentative n'a été faite à ce jour pour combiner le multiple local avec des scores ayant pour base un arbre connu ou estimé.

À nouveau ici, aucune des économies qu'il est possible de réaliser dans le calcul d'un score ou d'un alignement optimal global ne s'applique au cas local. La complexité en temps de l'agorithme est donc en $O(n^N)$ lorsque seul le meilleur score ou un meilleur alignement est recherché. La complexité en espace est quant à elle en $O(n^N)$ pour la recherche d'un meilleur alignement et en $O(n^{N-1})$ pour celui du meilleur score.

Ainsi que nous l'avons dit, une telle approche exhaustive n'est pas utilisée en pratique et n'a jamais en fait été formalisée comme dans le cas du deux à deux. Par contre, plusieurs heuristiques ont pour base de départ les principes de calcul de la programmation dynamique et ce sont elles que nous montrons dans la section suivante.

4.2.2.1.2 Les méthodes approchées

4.2.2.1.2.1 Les méthodes approchées pour du deux à deux — le cas particulier du criblage de banque

Les méthodes de comparaison locale dont nous allons parler d'abord font en réalité du deux à deux, dans un seul cas il y a comparaison de trois objets simultanément. Nous les plaçons malgré cela à cet endroit du chapitre car les algorithmes concernés ont à effectuer un très grand nombre de ces comparaisons deux à deux, et doivent ainsi faire face à des problèmes de temps d'exécution d'une amplitude similaire à ceux rencontrés par les algorithmes de comparaison multiple d'un ensemble de chaînes.

En effet, le sujet qui va nous concerner dans cette section peut être vu comme un cas un peu particulier d'une exploration de la ressemblance entre séquences macromoléculaires telle que présentée jusqu'ici. Désormais, parmi l'ensemble des chaînes qui doivent être traitées, une va se trouver privilégiée par rapport aux autres. Cette chaîne privilégiée (qui est appelée chaîne requête — en anglais 'query sequence') représente le plus souvent une macromolécule dont la séquence aura été récemment déterminée mais dont on ignore encore quelle fonction elle exerce et à quelle famille elle appartient. Afin d'essayer d'établir son 'identité' et le rôle qu'elle joue, elle est donc comparée à d'autres séquences pour lesquelles ces informations sont connues et qui se trouvent placées dans une banque. Or les banques actuelles contiennent une quantité considérable d'éléments (52 205 séquences pour les banques de protéines (SWISS PROT, version 33, Février 1996) et 701 246 pour les banques d'acides nucléiques (EMBL, version 46, Mars 1996), représentant 18 531 384 acides aminés et environ 473 000 000 bases nucléiques respectivement).

Il était jusqu'à récemment hors de question d'utiliser un algorithme de programmation dynamique (le plus sensible pour des comparaisons deux à deux) afin de réaliser toutes les comparaisons possibles entre la chaîne privilégiée et les chaînes de la banque. La complexité

d'une telle analyse, appelée un criblage de banque, est en $O(n.m.M)$ où n est la longueur de la chaîne requête, m est la longueur moyenne des chaînes de la banque et M est le nombre de celles-ci (observons que $M >> N$ où N est le nombre moyen de chaînes traitées par les algorithmes de comparaison multiple). Bien sûr, l'introduction d'implémentations parallèles de l'algorithme de programmation dynamique [Brutlag *et al.*, 1993] [Coulson *et al.*, 1987] a rendu ce problème moins aigu mais les techniques utilisées par les deux programmes de criblage de banque décrits ci-dessous présentent néanmoins un intérêt parce que les heuristiques qu'ils emploient ont souvent inspiré les algorithmes de comparaison locale de plusieurs chaînes que nous verrons plus loin, ou parce qu'ils utilisent une notion de ressemblance qui est intéressante et adaptable au cas multiple (définition 3.5.14). Tous deux peuvent être vus comme réalisant une approximation de la programmation dynamique à deux dimensions. La première méthode due à Pearson se ramène en fait à un simple comptage de monomères ou de k-mères identiques (qui peut être étendu au comptage de k-mères similaires par une relation R) tandis que la seconde (BLAST) effectue de la programmation dynamique sans trou à partir d'un certain nombre de 'noyaux de ressemblance' repérés au préalable par une technique basée sur la génération de 'synonymes'.

Nous commençons par décrire la plus élémentaire des deux méthodes, celle de Pearson.

Ce cribleur de banque élémentaire et relativement encore assez utilisé par la communauté biologique s'appelle FastN ou FastP suivant que les séquences traitées sont des séquences nucléiques ou des protéines, ou FastA dans sa dernière version pour protéines. Ce programme constitué ainsi par un ensemble de sous-programmes a été développé par Pearson et Lipman entre 1985 et 1988 [Lipman and Pearson, 1985] [Pearson and Lipman, 1988] [Pearson, 1990] [Pearson, 1991].

FastN/FastP et FastA reposent sur le même principe de fonctionnement. L'algorithme présente donc toujours une première phase de pré-traitement puis se divise en trois étapes. La première et la seconde sont communes aux deux programmes tandis que la troisième étape est propre à FastA. Nous ne décrivons ici que les deux premières qui forment l'essentiel de la méthode et esquissons seulement l'extension réalisée par FastA.

La phase de pré-traitement réalise la construction d'une simple table indexée des mots de longueur k (appelés des 'k-uples') de la chaîne requête, où k est une valeur fixée à l'avance (à 1 ou plus souvent 2 pour les protéines et à 4 ou 6 pour l'ADN/ARN). La taille de cette table est donc égale au nombre total de k-uples possibles, soit 4^k ou 20^k. À chaque k-uple est associé une valeur unique, c'est-à-dire un code identifiant l'entrée correspondante de la table et chaque entrée de la table contient la liste des positions sur la chaîne requête où ce k-uple particulier est rencontré.

La première étape de l'algorithme utilise alors une variante de la méthode des diagonales conçue par Wilbur et Lipman [Wilbur and Lipman, 1983] afin d'identifier ce qui est appelé les régions de similarité entre la chaîne requête r et chacune des chaînes b de la banque. Chacune de ces régions correspond à un glissement de r par rapport à b (voir la figure 4.12), puis à un alignement local, sans trous, des deux chaînes commençant et se terminant par un mot identique de taille supérieure ou égale à k. À l'intérieur des régions alignées peuvent alterner des mots identiques et d'autres qui ne le sont pas. Le nombre de ces derniers ne peut néanmoins dépasser une certaine valeur l fixée à l'avance (voir la figure 4.13). Parmi toutes ces régions de similarité locale, les dix ayant la plus grande densité, c'est-à-dire le plus grand nombre de k-uples identiques avec la chaîne requête, sont conservées (dans le cas où $k > 1$, les k-uples peuvent se chevaucher). Ceci revient à choisir les dix meilleurs alignements locaux, ou glissements. Tous les alignements sans trous possibles de la chaîne requête r avec une des chaînes b de la banque peuvent être visualisés à travers une matrice dont les colonnes représentent les symboles de

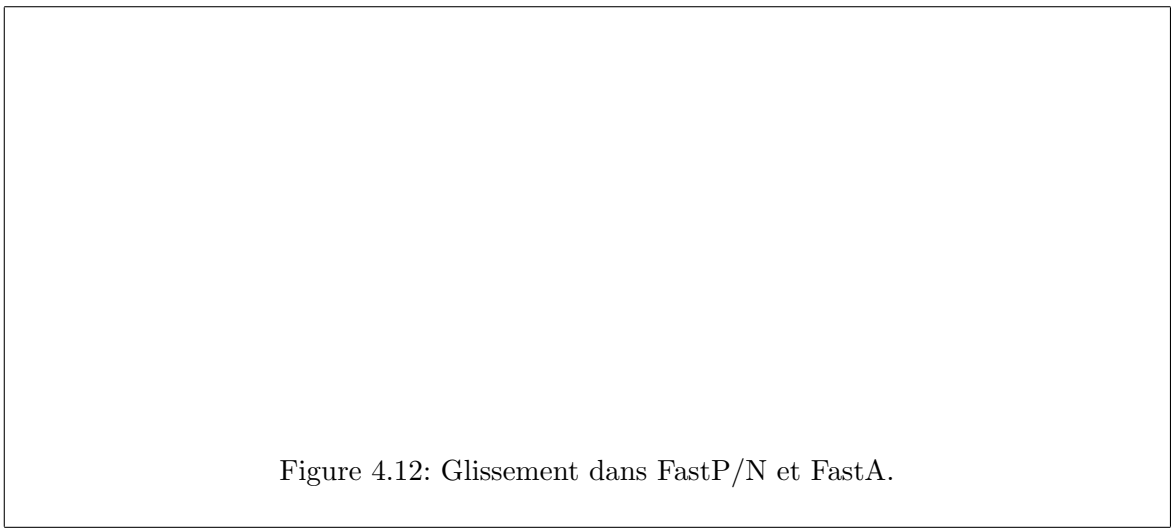


Figure 4.12: Glissement dans FastP/N et FastA.

la chaîne banque et les lignes celles de la requête. Cette matrice que nous dessinons dans la figure 4.13 n'est bien sûr pas effectivement construite. Dans l'algorithme, seul est utilisé un tableau des diagonales de la matrice (d'où le nom de la méthode) dont chacune correspond à un glissement de r par rapport à b. L'entrée d du tableau indique ainsi le nombre de k-uples (éventuellement chevauchants) appartenant à la dernière région de l'alignement obtenu par un décalage de d positions d'une chaîne relativement à l'autre. Par ailleurs, une file de priorité sert à conserver en permanence les informations permettant de repérer les dix meilleures régions (celles contenant le plus grande nombre de k-uples), à savoir leurs positions de début et fin sur chacune des chaînes.

La seconde étape consiste alors à utiliser une matrice de substitution, telle que PAM250, pour attribuer un score aux dix régions sélectionnées à la première étape. Ainsi le choix des régions à conserver est effectué sur la base d'un comptage de k-uples identiques (éventuellement séparés entre eux par des k-uples différents ainsi que nous l'avons vu) mais leur 'valeur' finale dépend de leurs scores dans le sens classique employé en programmation dynamique (sans trou - voir la definition 3.2.17). Observons que ce score porte en outre sur les k-uples identiques et sur les mots compris entre ces derniers dont le nombre est inférieur à l (c'est-à-dire, il porte sur toute une région). Un tri des régions d'après leur nombre de k-uples identiques peut ainsi ne pas être le même que celui obtenu d'après ces scores. Dans le cas de FastN/FastP, la ressemblance d'une chaîne de la banque par rapport à la chaîne requête est alors mesurée par le meilleur de ces scores.

La troisième étape de FastA consiste simplement à vérifier s'il n'est pas possible de joindre quelques-unes parmi les dix régions initialement identifiées afin de former un seul alignement ayant un score supérieur à celui des dix pris séparément. FastA essaye ainsi de réaliser une sorte d'alignement avec trous — les trous correspondant cependant aux segments de jonction entre deux régions, et seulement à ces segments. Les régions initiales sont préservées telles quelles, aucun trou n'étant permis à l'intérieur. Pour effectuer cet alignement, FastA emploie une technique de programmation dynamique [Pearson, 1990] et c'est le score de cette région qui, dans le cas de FastA, mesure la ressemblance de chacune des chaînes de la banque avec la chaîne requête.

Le résultat de l'algorithme quelle que soit la version consiste alors en la présentation d'une

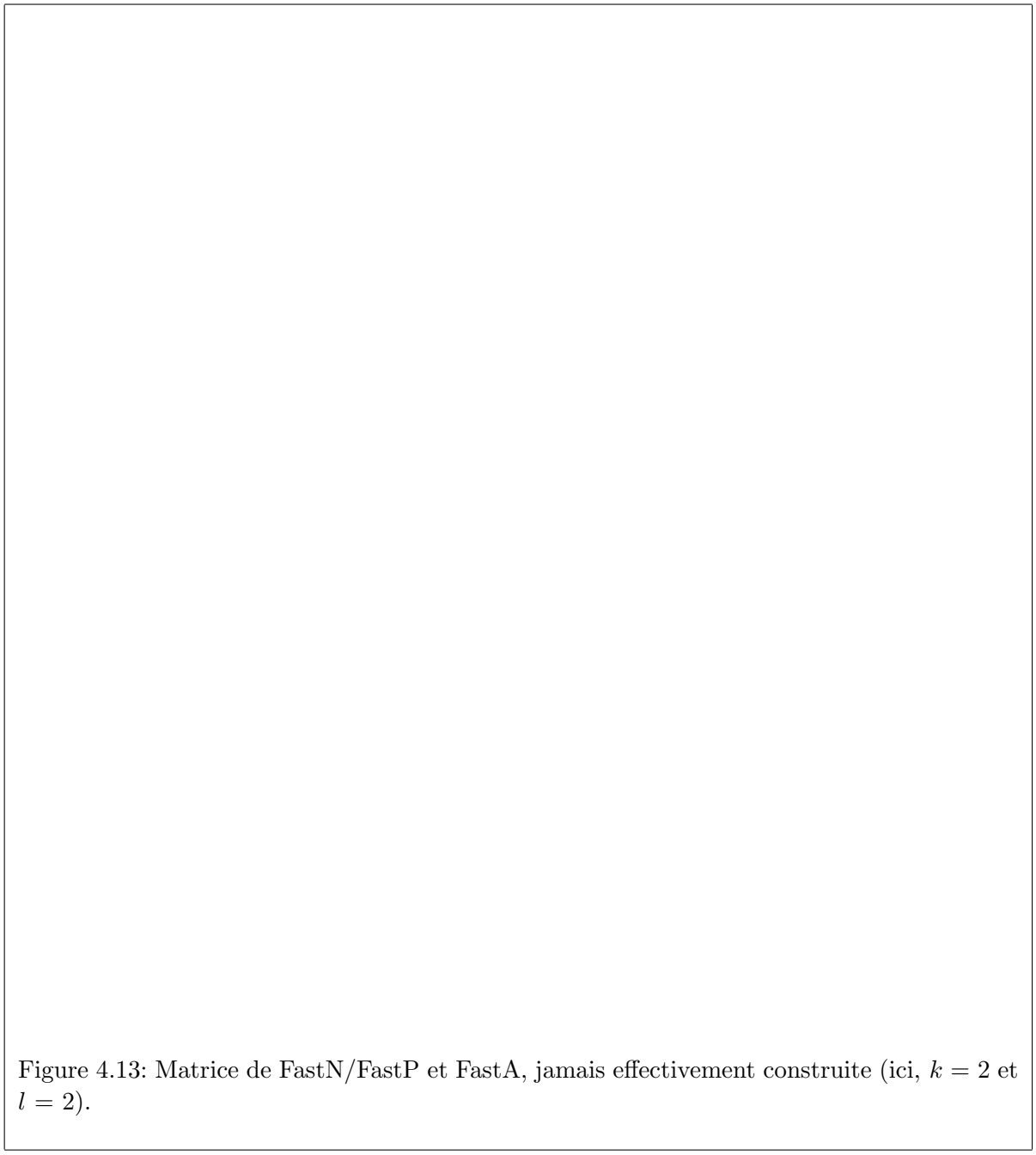


Figure 4.13: Matrice de FastN/FastP et FastA, jamais effectivement construite (ici, $k = 2$ et $l = 2$).

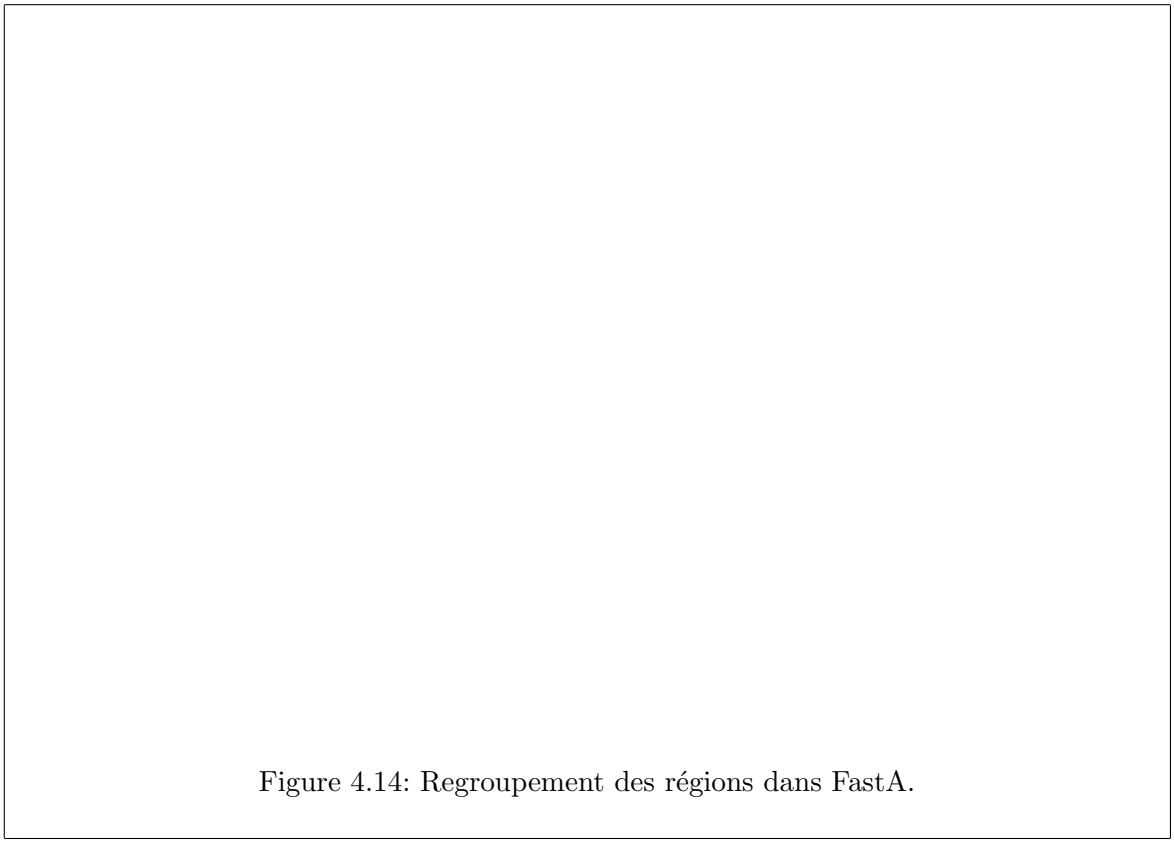


Figure 4.14: Regroupement des régions dans FastA.

```
/* Construction de la table indexée */
/* Entrée */
        n = longueur de la chaîne requête
        k = 2 dans cet exemple
/* Structure de données */
        V = tableau de taille | Σ |^k de pointeurs sur une liste de positions de la chaîne
        requête (V[i] pointe sur la liste de toutes les occurrences des mots de longueur k
        de la chaîne requête en relation de similarité avec l'élément de Σ^k codé par l'entier
        i)

        pour chaque facteur r_i r_{i+1} situé à la position i de la chaîne requête
              pour (a ∈ Σ)
                    si (a R r_i)
                          pour (b ∈ Σ)
                                si (b R r_{i+1})
                                      placer la position i dans la liste V[code(ab)];
```

Figure 4.15: Phase de pré-traitement de FastN/FastP et FastA modifiés par Viari : Construction de la table indexée pour une relation de similarité R^k.

liste triée par les scores en ordre décroissant de toutes les chaînes de la banque pour lesquelles ce score est supérieur à un certain seuil fixé par l'utilisateur.

Comme nous pouvons le constater, ces programmes sont très simples mais relativement efficaces pour repérer des ressemblances lorsque celles-ci ne sont pas trop faibles. Il est également aisé d'introduire plus de souplesse dans la recherche en travaillant non plus avec une relation d'identité entre k-uples mais avec une relation de similarité.

Une première extension de l'algorithme a ainsi été réalisée par Brutlag [Brutlag *et al.*, 1990] pour laquelle deux k-uples $u_1...u_k$ et $v_1...v_k$ sont similaires si :

$$\mathcal{M}(u_1, v_1) \times \mathcal{M}(u_2, v_2) \times ... \times \mathcal{M}(u_k, v_k) \geq t^k$$

où $\mathcal{M}$ est une matrice de similarité et t est une valeur seuil.

Il est possible également d'établir une relation de similarité entre k-uples à partir d'une relation de similarité entre monomères comme celle décrite dans la section 3.4.3.2 du chapitre 3 [Viari and Pothier, 1993]. Dans ce cas, les k-uples $u_1, ..., u_k$ et $v_1, ..., v_k$ sont dits similaires si $u_i \ R \ v_i$ pour tout $i \in \{1, ..., k\}$.

En termes de l'algorithme, seule change la phase de pré-traitement, à savoir la construction de la table indexée à partir de la chaîne requête r. Celle-ci est évidemment plus remplie et nécessite plus de temps pour être parcourue. Une idée de l'algorithme pour cette phase est donnée dans la figure 4.15. La recherche des k-uples similaires entre r et chacune des chaînes de la banque puis la sélection des meilleures régions se fait de la même façon que pour FastN/FastP et FastA.

Voyons maintenant quelle est la complexité du programme de Pearson. La construction de la table indexée est au pire en $O(n. \mid \Sigma \mid^k)$ où n est la longueur de la chaîne requête. La recherche des k-uples quant à elle se fait en un temps $O(\frac{n.m.M.g^k}{|\Sigma|^k})$ où m est la longueur moyenne des

chaînes de la banque et M est leur nombre, et avec bien sûr $g = 1$ pour les versions originales de FastN/FastP et FastA.

Nous passons maintenant au second programme de criblage de banque, BLAST, de loin le plus connu et utilisé. BLAST ('Basic Local Alignment Search Tool') a été développé par Altschul et ses collègues [Altschul *et al.*, 1990] comme une alternative à FastN/FastP et FastA. La philosophie suivie par ce programme est celle qui considère la valeur optimale d'une fonction de score comme étant la meilleure mesure de la ressemblance entre deux chaînes. Idéalement il faudrait donc pouvoir repérer ce que les auteurs appellent les 'paires de segments maximales' (c'est-à-dire de mots) entre la chaîne requête et chacune des chaînes de la banque. Une paire de segments est dite maximale (abrégé en MSP — en anglais 'Maximal Segment Pair'), si elle possède le plus grand score parmi toutes les paires de segments de longueur identique appartenant aux deux chaînes, où le score d'une paire est calculé à l'aide d'une matrice de similarité (voir la section 3.2.2.2). Les MSP correspondent donc tout simplement à ce que localise l'algorithme de programmation dynamique appliqué au cas local de Smith et Waterman (sans trou) [Smith and Waterman, 1981] (voir la section 4.2.2.1.1).

Bien sûr, afin de trouver les MSP de deux chaînes de manière exacte, il faudrait faire appel à cet algorithme dont la complexité en temps est proportionnelle au produit des longueurs des chaînes. Afin d'aller plus vite, il est alors nécessaire d'employer une heuristique. Celle de BLAST repose sur une observation très simple. Lorsque l'on effectue une recherche dans une banque contenant des milliers de chaînes, il arrive fréquemment que seul un petit nombre de ces chaînes, parfois même aucune, présente une similarité significative avec la chaîne requête. BLAST accélère donc la recherche dans une banque en minimisant le temps passé à analyser des chaînes dont le score de la MSP qui serait obtenue avec la chaîne requête a peu de chances d'excéder une certaine valeur S dont nous verrons plus loin comment elle est établie. Pour ce faire, il va considérer que les chaînes qui peuvent posséder une MSP avec un score supérieur à S sont uniquement celles qui contiennent au moins un mot de longueur w ayant un score d'au moins $T < S$ lorsque aligné, sans trous, avec un mot de la requête.

BLAST effectue ainsi initialement un simple filtrage de la banque qui a pour but d'éliminer toutes les chaînes ne contenant aucun mot de longueur w ayant un score d'au moins T avec un mot de la chaîne requête. Pour cela, il commence par considérer tous les mots de longueur w de la chaîne requête et établit une liste qui, pour chacun d'eux, indique les éléments de Σ^w dont le score avec le mot de la requête excède T. Les mots qui sont placés dans cette liste sont appelés des w-mères. La liste finale peut représenter un volume important (typiquement plus de 10000 mots) mais elle peut être produite en un temps directement proportionnel au nombre d'éléments qu'elle contient. Observons également qu'un mot de la chaîne requête peut être représenté par plusieurs mots dans la liste, ou, au contraire, par aucun. En effet, rappelons que dans certaines matrices de similarité (par exemple la matrice PAM250 donnée dans la figure 3.3) la substitution d'un acide aminé par lui-même ne possède pas toujours le score le plus élevé. Aussi, suivant la valeur fixée pour T, certains mots de la chaîne requête peuvent avoir un score en-dessous de T lorsqu'ils sont alignés avec eux-mêmes. Enfin la liste peut être préservée sous forme d'une table indexée ou sous forme d'un automate minimal. Quel que soit son mode de représentation, une fois cette liste construite, une recherche séquentielle de chacune des chaînes de la banque permet de décider rapidement si celle-ci contient un mot de longueur w qui forme une paire avec un mot de la chaîne requête ayant un score supérieur ou égal à T. Si cela est le cas, l'adresse de ces mots (appelés des 'hits') est conservée pour la deuxième phase de l'algorithme.

Dans cette seconde phase, BLAST essaye de prolonger (à gauche et à droite) tout mot ('hit') ainsi trouvé sur une des chaînes de la banque afin de déterminer si la paire de score supérieur à

T qu'il forme avec un des mots de la chaîne requête n'est pas contenue dans une paire de mots plus longs possédant un score supérieur ou égal à S. Le prolongement des mots initiaux se fait également sans trous. Il est clair que plus petite est la valeur choisie pour T, plus grande est la probabilité qu'une paire de mots ayant un score d'au moins S contienne une paire de mots ayant un score d'au moins T et ne soit donc pas 'ratée'. Cependant, une petite valeur pour T a aussi pour effet d'augmenter le nombre de 'hits', et donc le temps d'exécution de l'algorithme. Par conséquent, un compromis est nécessaire, en général T est choisi égal à environ un tiers de la valeur de S. Enfin, le processus d'allongement dans une des directions est arrêté dès que le score de substitution de la paire de symboles que l'on essaye de rajouter aux mots déjà alignés est inférieur à une valeur X fixée par l'utilisateur.

Le programme garde toutes les chaînes de la banque possédant au moins un 'hit' qui a pu être prolongé jusqu'à être contenu dans une paire de mots de score au moins S et attribue à chacune de ces chaînes le score du meilleur prolongement. Il calcule également la 'p-value' correspondant à ce score, c'est-à-dire, la probabilité avec laquelle un score supérieur ou égal aurait pu être atteint sur une chaîne aléatoire de même longueur et de même composition en monomères. Le calcul de cette probabilité n'est pas réalisé de manière empirique par une comparaison avec de l'aléatoire comme cela est le cas pour FastP/N et FastA mais aussi pour d'autres programmes de criblage moins connus [Collins *et al.*, 1988] [Collins and Coulson, 1990], mais est effectué *a priori*, sur la base d'une théorie statistique établie par Karlin [Altschul, 1991] [Karlin and Altschul, 1990]. Ainsi, cette p-value est donnée par la formule :

$$1 - \ exp^{-Knm\ exp^{-\lambda S}}$$

où K est un facteur de normalisation qui dépend uniquement de la matrice de similarité $\mathcal{M}$ utilisée et λ est donné par $\sum_{i,j\in\Sigma} p_i p_j \exp^{\lambda \mathcal{M}(i,j)} = 1$ avc p_i la fréquence d'apparition du monomère i dans la banque.

C'est cette même formule qui est bien sûr employée dans l'autre sens afin d'obtenir la valeur à attribuer à S, et qui doit correspondre à la valeur du plus grand score que peut présenter une MSP relativement à deux chaînes prises au hasard (en général, celle correpondant à une p-value de 10^{-3} ou 10^{-5}). Enfin, le classement des chaînes de la banque est effectué sur les 'p-values' plutôt que sur les scores. L'emploi de ces p-values afin de d'estimer le degré de ressemblance de la chaîne requête avec les chaînes de la banque, ainsi que l'utilisation d'une unité de comparaison ayant pour base des mots (les w-mères) plutôt que des symboles font toute l'originalité de BLAST, et son importance en pratique pour les biologistes. Quant à l'algorithme lui-même, son intérêt principal réside dans l'automate des w-mères dû à Myers [Altschul *et al.*, 1990].

Certaines étapes de BLAST admettent des variantes. Par exemple, dans la phase de prolongement des 'hits', des trous peuvent être permis. Une forme simplifiée de programmation dynamique est alors utilisée pour effectuer ce prolongement, mais il est clair que cela augmente considérablement le temps d'exécution de l'algorithme et n'est pas une variante vraiment utilisée en pratique. Une autre modification possible consiste à créer la liste des mots à partir des chaînes de la banque de protéines et de rechercher ensuite des 'hits' dans la chaîne requête. Bien sûr, cela demande un espace en mémoire considérable, et exige un accès aléatoire à la banque et, encore une fois, est une variante peu employée.

Une extension bien plus intéressante de BLAST, appelée BLAST3, permet de comparer trois chaînes de la banque simultanéement et non plus deux. La comparaison toutefois n'est pas faite directement mais utilise les paires de mots requête-banque ayant un score suffisament élevé (typiquement supérieurs à une valeur $T' < S$) qui auront été repéré au préalable. Ces paires permettent alors d'établir les alignements, toujours sans trous, de triplets de mots, un mot de la requête et deux de la banque appartenant à des chaînes différentes. Ainsi, si le mot v

sur la chaîne requête r est aligné avec les mots $u1$ et $u2$ de deux chaînes $s1$ et $s2$ de la banque, alors le trio v, $u1$ et $u2$ détermine un certain alignement de r, $s1$ et $s2$, que BLAST3 essaye de prolonger de façon similaire au prolongement des paires dans BLAST (le score de trois symboles par lequel ce prolongement est tenté étant donné par un score SP : c'est la somme des scores des trois paires de symboles qu'il est possible d'obtenir à partir du trio de départ). À nouveau ici, le choix de la valeur T' est un compromis entre sensibilité et rapidité. Les triplets de mots de score supérieur à T' sont quant à eux repérés comme le sont ceux de score supérieur à S dans BLAST (c'est-à-dire, en deux phases), ou bien directement à partir de la table indexée ou automate de la première étape de filtrage de BLAST (localisation des 'hits').

Il est plus difficile d'établir la complexité de BLAST (ou de BLAST3) puisque celle-ci va dépendre des valeurs attribuées à tous ces paramètres, S, T, X (et T' dans le cas de BLAST3). Dans la pire des hypothèses (pour $S = T = T' = 0$ et X égal à la valeur maximale de la matrice de similarité), elle est en $O(n.m.M)$ pour BLAST et $O(n.(m.M)^2)$ pour BLAST3 comme pour les algorithmes de programmation dynamique. En pratique, BLAST est cependant environ 10 fois plus rapide que FastA, dont nous avons vu que la complexité est en $O(\frac{n.m.M}{|\Sigma|^k})$.

4.2.2.1.2.2 Les méthodes approchées pour le multiple

Nous allons passer maintenant au cas multiple et présenter dans cette section quatre heuristiques, dont les trois premières ne sont qu'esquissées. Il convient d'observer dès maintenant qu'aucun de ces quatre algorithmes n'admet de trous dans les blocs de similarité qu'ils identifient.

Les deux premières heuristiques que nous décrivons rapidement procèdent en fixant à l'avance la longueur des similarités locales qu'elles vont pouvoir repérer. Notons cette longueur m. Les ensembles de mots similaires sont alors recherchés soit en faisant de la programmation dynamique à N dimensions mais uniquement à l'intérieur d'une fenêtre de longueur $W > m$ [Johnson and Doolittle, 1986], soit en effectuant des comparaisons progressives [Bacon and An- derson, 1986] [Bacon and Anderson, 1990]. Dans ce second cas, les mots similaires sont localisés en réalisant des comparaisons deux à deux toujours. Initialement deux chaînes sont choisies (sur la base de critères non spécifiés dans l'article) et tous les mots similaires de longueur m recherchés, classés suivant leurs scores et puis placés dans une table. Une troisième chaîne est ensuite choisie (à nouveau les critères pour ce choix ne sont pas donnés) et comparée avec la table des mots similaires repérés sur les deux chaînes initiales. Cette comparaison résulte en une nouvelle table, plus réduite que la première, contenant les mots similaires communs aux trois chaînes. Le processus se poursuit avec les chaînes restantes qui sont comparées à tour de rôle à la table des mots similaires présents sur les chaînes examinées auparavant pour produire à la fin une table des mots similaires communs à toutes. La complexité en temps de ces algorithmes est en $O(N.(n - W).W^{N-1})$ pour Johnson et $O(n^2.N^2.m.M)$ où M est la taille de la table (de 'hachage') pour Bacon, n et N la longueur moyenne et le nombre de chaînes respectivement. Un inconvénient majeur de ces deux approches est bien sûr que longueur des similarités locales doit être fixée *a priori*. Par ailleurs, la recherche de ces similarités n'explore pas tout l'espace possible, soit parce qu'elle n'est en fait réalisée qu'à l'intérieur d'une fenêtre de longueur également fixe (Johnson), soit parce qu'elle limite le nombre de chaînes qui sont effectivement comparées entre elles (Bacon). Les mots similaires trouvés par ce dernier dépendent également fortement de l'ordre avec lequel les chaînes ont été choisies pour être comparées.

La troisième heuristique due à Sheridan [Sheridan and Venkataraghavan, 1992] a pour point de départ une recherche des mots similaires communs à deux ou trois chaînes et utilise pour trouver ces mots le cribleur BLAST [Altschul *et al.*, 1990] et sa version étendue BLAST3

[Altschul and Lipman, 1990] décrits précédemment (section 4.2.2.1.2.1). L'extension de Sheridan au cas de plus de 3 chaînes fait appel à une technique de regroupement (en anglais 'clustering') sur une sélection des 'meilleurs' ensembles de deux ou trois mots similaires (ceux ayant un meilleur score total et, dans le cas des triplets, deux à deux) identifiés dans la première phase grâce à BLAST3, mais qui est différente de celle employée par BLAST3, pour mettre ensemble les paires de mots trouvés dans une étape préliminaire. En effet, le critère pour grouper des duplets ou triplets de mots similaires est cette fois que p des mots doivent avoir un préfixe ou un suffixe de longueur au moins m en commun, où m est typiquement choisi égal à 15 et p à 1 (pour le regroupement de deux duplets) ou 2 (pour le regroupement d'un duplet avec un triplet ou de deux triplets). La complexité de l'algorithme n'est pas donnée mais l'emploi de BLAST3 signifie que, dans le pire des cas, elle est en $O(n^3)$.

Enfin, la quatrième heuristique que nous allons décrire plus en détail est le premier algorithme à avoir été inclus dans un programme appelé MACAW [Schuler *et al.*, 1991] (pour simplifier, nous confondrons dorénavant le nom de cet algorithme avec celui du programme). Cet algorithme est très utilisé en pratique et se libère de presque toutes les limitations des heuristiques antérieures. La longueur des similarités locales n'a ainsi pas besoin d'être établie au préalable et les comparaisons sont faites simultanément et non progressivement comme dans le cas de Bacon. Par contre, les trous ne sont toujours pas autorisés dans les ensembles de mots considérés comme similaires. Ceux-ci sont donc soit des mots identiques, soit des mots qui présentent, les uns par rapport aux autres, un certain nombre de substitutions uniquement. L'espace de recherche n'est pas non plus exploré exhaustivement et l'idée d'une progression demeure dans MACAW puisque les mots similaires communs à $p + 1$ des chaînes de départ sont construits en utilisant les mots similaires communs à 2 chaînes obtenus dans une étape initiale. Mais toutes les combinaisons deux à deux des N chaînes sont tentées afin d'en extraire les mots communs, ce qui n'est pas le cas pour Bacon. En fait, MACAW est très ambitieux puisqu'il souhaite repérer non seulement les mots similaires présents dans toutes les chaînes, mais aussi ceux qui ne sont présents que dans un sous-ensemble de celles-ci. Il veut donc pouvoir identifier tous les groupes de mots communs, même lorsque ceux-ci ne contiennent que 2 éléments. C'est une caractéristique intéressante de l'algorithme, nous avons en effet vu dans la section 3.3.3 qu'il est important de pouvoir travailler en présence de bruit.

L'idée de l'algorithme est la suivante. Commençons par le deux à deux. MACAW emploie de la programmation dynamique pour comparer deux chaînes localement mais comme il n'autorise pas les trous, nous pouvons voir les calculs effectués comme un simple comptage, pondéré par des scores, des symboles mis face à face lorsqu'une chaîne $s1$ est alignée avec une seconde chaîne $s2$, pour tout glissement possible de $s1$ par rapport à $s2$. Comme dans le cas de FastN/FastP ou FastA et BLAST, ceci signifie que dans le tableau des scores, seul nous intéresse ce qui se passe le long des diagonales puisque celles-ci représentent tous les alignements possibles sans trous des deux chaînes (voir la figure 4.16).

Pour chaque diagonale, MACAW recherche alors les paires de mots similaires définis comme étant ceux qui ont un score supérieur à une certaine valeur seuil t. Ces paires forment ce que les auteurs appellent des blocs de quorum en chaîne 2 (voir la définition 3.3.4). Idéalement, pour rechercher tous les blocs de quorum en chaîne 3 à N de score supérieur à ce seuil (où le score d'un bloc multiple est la somme des scores de toutes les combinaisons possibles de paires de mots du bloc, c'est-à-dire c'est une somme SP), la procédure serait la même. Elle consisterait donc en un examen de toutes les diagonales d'un tableau ayant de 3 à N dimensions. Une telle démarche conduirait cependant à un algorithme dont la complexité serait en $O((n+1)^{N+1})$, ce qui est pire que de la programmation dynamique classique à N dimensions. L'heuristique qui permet alors à MACAW de demeurer en $O(n^2.N^2)$ repose sur le même principle que celui

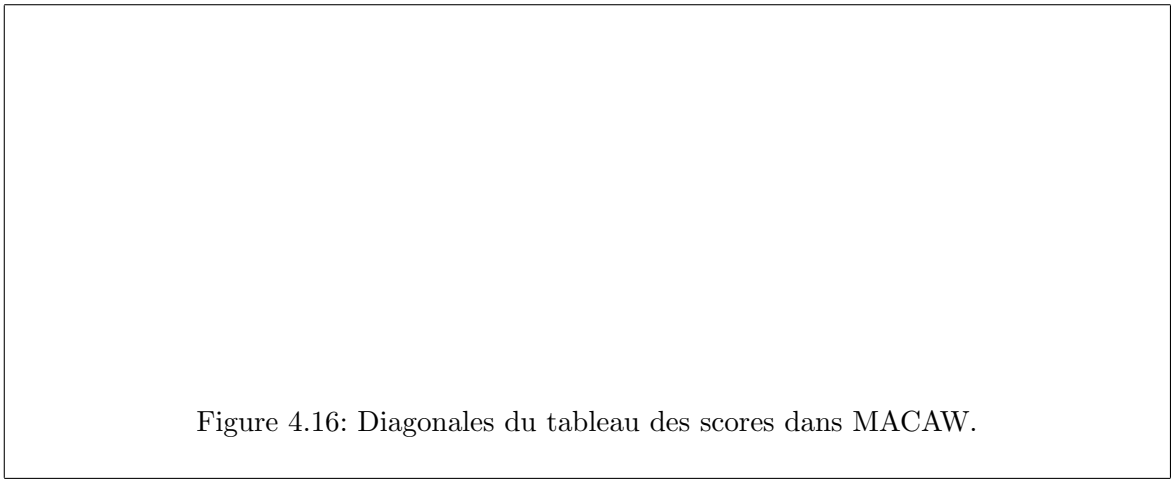


Figure 4.16: Diagonales du tableau des scores dans MACAW.

utilisé par BLAST3, étendu cette fois au cas de N chaînes. En effet, rappelons que les blocs de quorum supérieur ou égal à 2 que MACAW veut retenir sont ceux qui ont un score supérieur à une certaine valeur seuil t. Considérons maintenant un bloc de quorum p strictement supérieur à 2. Il est tout à fait possible (cela sera même souvent le cas) qu'une ou plusieurs des paires de mots de ce bloc n'aient pas un score supérieur à t même si le bloc comme un tout vérifie cette propriété puisque le score d'un bloc est la somme des scores de toutes les paires de mots le composant. L'idée est alors d'imposer une contrainte supplémentaire assez forte qui consiste à exiger qu'un bloc de quorum $p \geq 3$ ait non seulement un score supérieur à t, mais que de plus il soit composé de paires de mots dont chacune a elle-même un score supérieur à t. Ainsi les blocs de quorum $p \geq 3$ sont construits uniquement à partir des blocs de quorum 2 qui auront été retenus dans une première étape de l'algorithme (voir la figure 4.17). MACAW ne recherche donc pas tous les blocs de quorum $p \geq 3$ dont le score serait supérieur à t, mais seulement un sous-ensemble (en pratique souvent considérablement plus petit) de ces blocs. La justification pour une telle contrainte est que les blocs multiples contenant des paires de mots qui ne se ressemblent pas beaucoup ne sont pas intéressants biologiquement. C'est une remarque valable, mais la 'solution' proposée semble un peu drastique dans la mesure où le seuil est fixé à la même valeur quel que soit le quorum du bloc (qu'il comporte donc 2 ou plus de 2 mots) ce qui n'était pas le cas dans BLAST3. On peut se demander si MACAW n'ouvre pas ainsi main en partie de la flexibilité que le multiple permet par rapport au deux à deux (voir la section 3.3.2).

Ce seuil t que doit dépasser le score de tout bloc est déterminé encore une fois sur la base d'un critère statistique, similaire à celui employé par BLAST : c'est le score au-delà duquel un bloc de quorum 2 est en fait considéré comme étant 'significatif', c'est-à-dire comme non dû au hasard. Par ailleurs, une seconde sélection peut être effectuée même sur les blocs de quorum $p > 2$. Ceux-ci ne sont alors retenus que s'ils sont eux-mêmes significatifs. La signification statistique d'un bloc de quorum p quelconque ayant un score égal à $\mathcal{S}$ est une extension au cas p de la théorie pour 2 chaînes et est donnée par la formule [Karlin and Altschul, 1993] [Schuler *et al.*, 1991] :

$$1 - \exp^{-K.n^p.\exp^{-\lambda.\mathcal{S}}}$$

avec ici K et λ calculées d'après les scores possibles pour une p-colonne (c'est-à-dire une colonne ayant p lignes/symboles) et les probabilités correspondant à ces scores. Cette formule suppose que le score moyen d'une colonne est négatif et que les longueurs des p chaînes où se trouve

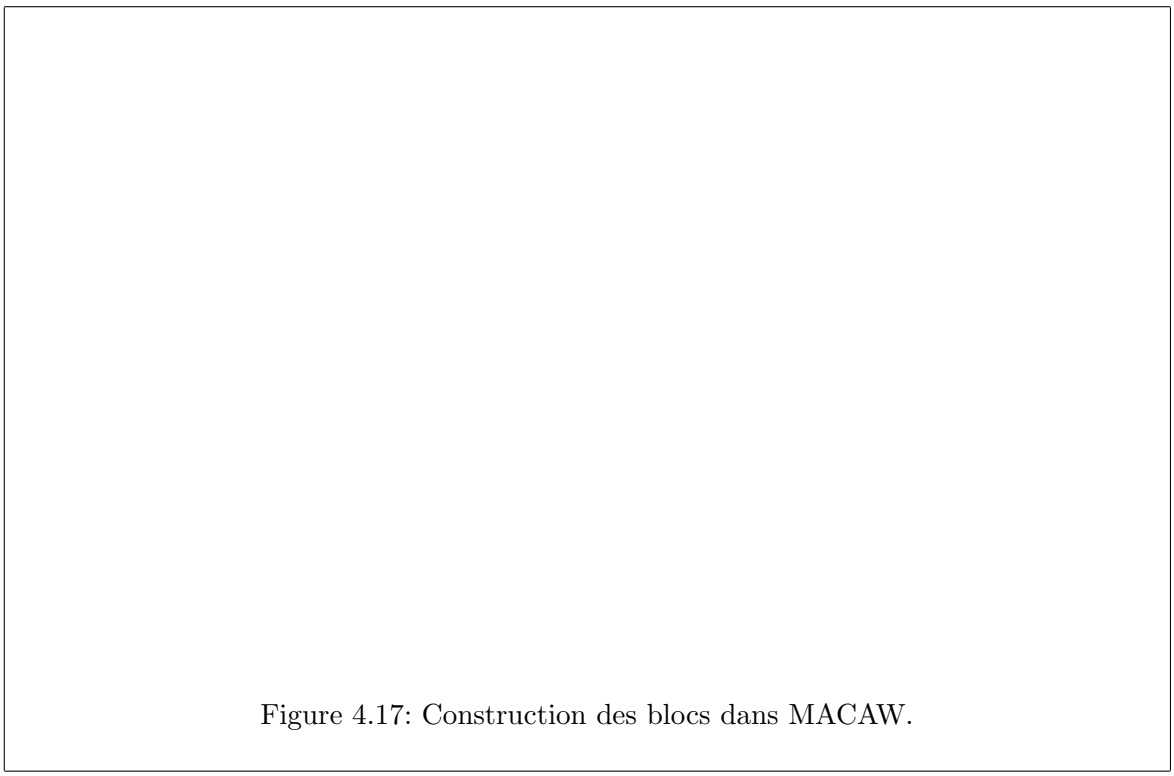


Figure 4.17: Construction des blocs dans MACAW.

le bloc ne sont pas trop différentes. Les blocs de quorum 2 ou plus peuvent ainsi être classés d'après la valeur de cette formule et seuls les plus 'significatifs' conservés. Ce classement permet alors à MACAW de réaliser un alignement multiple des chaînes. Pour cela, il sélectionne le bloc le plus significatif de tous (notez qu'il peut y en avoir plus d'un) et aligne sur lui toutes les chaînes où ce bloc est présent. Puis le second bloc le plus significatif est choisi, et s'il ne présente pas de conflit avec le premier (c'est-à-dire, si ses occurrences ne se situent pas à la fois à gauche et à droite du premier bloc), les chaînes où il est présent sont alignées sur ce second bloc (avec création éventuelle de trous parfois très étendus entre les blocs). Le processus se poursuit tant que cela est possible pour tous les blocs considérés comme étant suffisamment significatifs (voir les figures 4.18 et 4.19 pour l'alignement final).

Il faut cependant noter que le calcul de K et de λ, qui cette fois est fonction du nombre de chaînes et des probabilités correspondant aux scores des colonnes des blocs, est très coûteux. Il est donc en pratique réalisé une fois pour toutes au préalable d'après des scores estimés sur la base des fréquences observées habituellement dans les macromolécules pour chacun des monomères, cela pour un nombre de chaînes variant entre 2 et une valeur maximale qui actuellement est fixée à 16. MACAW ne peut donc pour l'instant repérer des blocs de similarité de manière fiable selon les principes qu'il a lui-même établis que sur un ensemble contenant au plus 16 chaînes.

Le grand intérêt de MACAW est que le programme est suffisamment rapide pour pouvoir être interactif. Un utilisateur peut ainsi éventuellement rejeter un bloc sélectionné par l'algorithme comme étant statistiquement significatif, ou plus simplement l'éditer, c'est-à-dire entre autres, étendre ou diminuer la longueur d'un bloc (i.e. longueur des mots le composant) ou son quorum (i.e. nombre de chaînes que le bloc englobe).

Signalons enfin qu'une technique initialement un peu semblable à celle de MACAW est

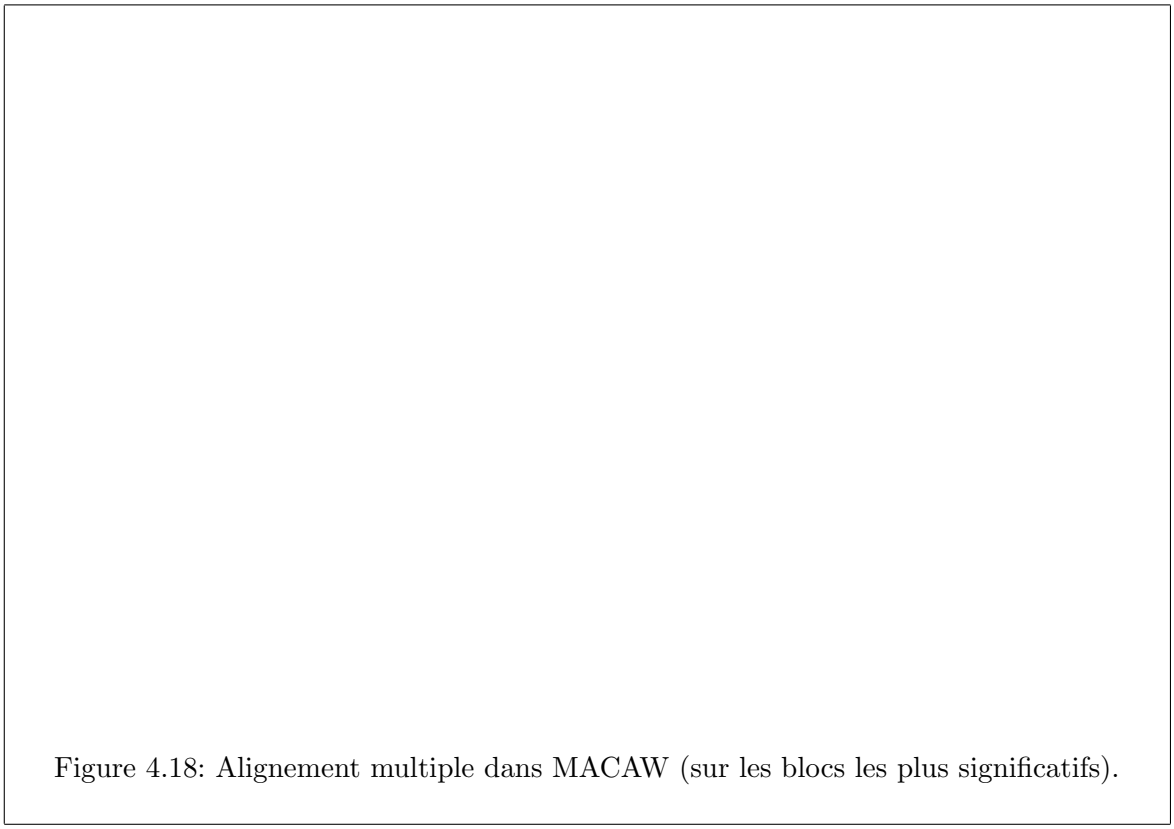


Figure 4.18: Alignement multiple dans MACAW (sur les blocs les plus significatifs).

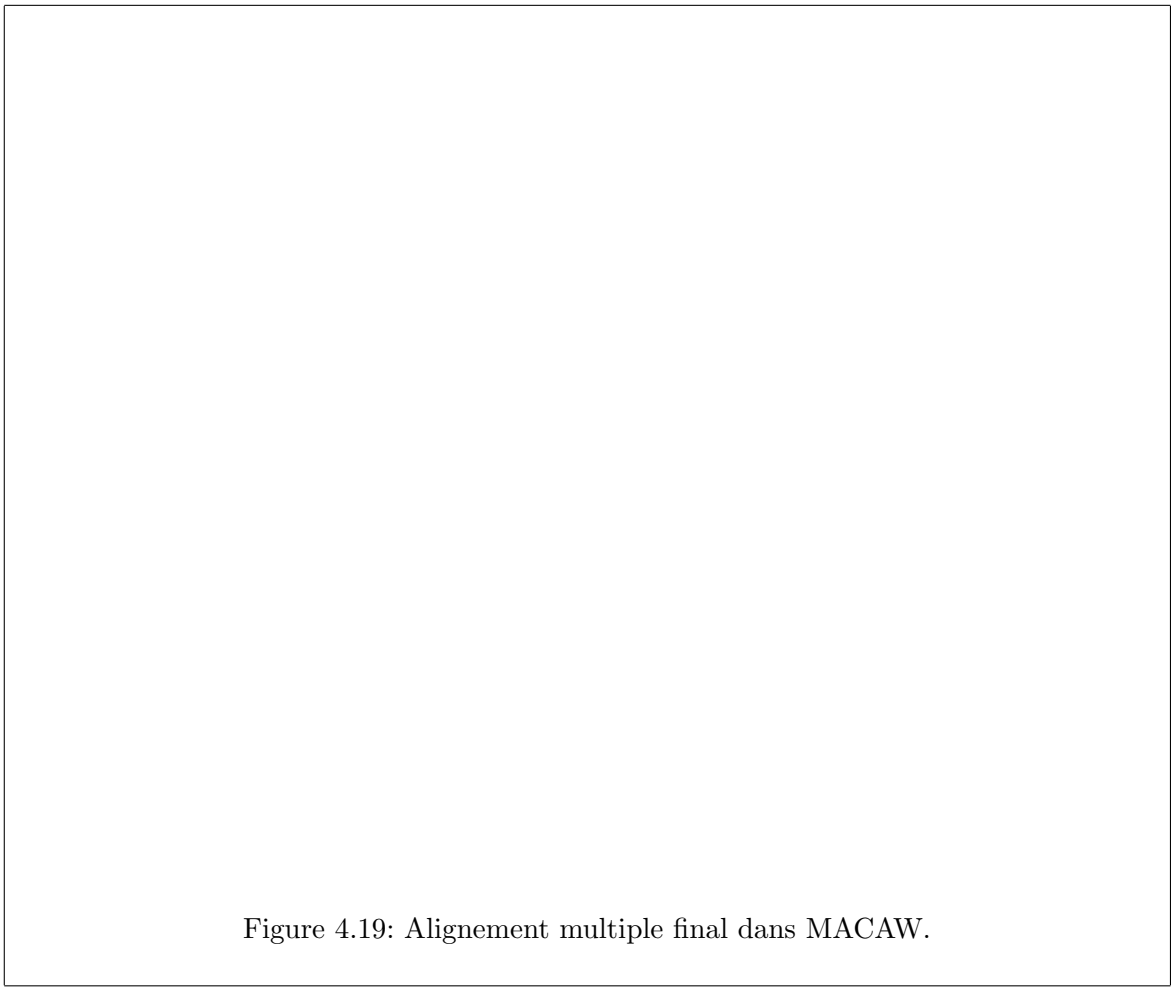


Figure 4.19: Alignement multiple final dans MACAW.

utilisée par Heringa [Heringa, 1994] [Heringa and Argos, 1993] afin de localiser des blocs de mots similaires (avec trous autorisés cette fois). La conservation d'un bloc se fait toutefois désormais sur la base de son score uniquement et il n'y a pas évaluation de la signification statistique des blocs.

4.2.2.2 Méthodes d'optimisation stochastique

Nous ne présentons dans cette section qu'une des méthodes d'optimisation stochastique, celle de Lawrence inspirée par l'approche de Hertz [Hertz *et al.*, 1990] [Cardon and Stormo, 1992] et introduite initialement en 1990 avec l'algorithme EM (en anglais 'Expectation Minimisation') [Lawrence and Reilly, 1990] et puis affinée en 1993 avec la méthode d'échantillonnage de Gibbs [Lawrence *et al.*, 1993]. Nous ne parlerons donc pas de l'autre méthode utilisant des HMM ('Hidden Markov Model') [Krogh *et al.*, 1994]. La raison en est que le principe de base est essentiellement le même. Il s'agit en effet toujours d'extraire une information (localisation d'un signal) d'une masse de données (les chaînes) d'après des critères statistiques (EM dans le cas de Krogh par exemple). Les différences fondamentales entre les deux approches sont dans le type de critère statistique utilisé et dans le fait que Lawrence condense l'information extraite dans une matrice de fréquences, tandis que les méthodes HMM la résument dans des automates stochastiques. Ces différences, en particulier quant au critère utilisé, ne sont pas négligeables, mais il n'est pas essentiel de les détailler dans le cadre de ce travail.

Il est intéressant avant de décrire l'algorithme de Lawrence dans sa seconde version (celle concernant l'échantillonnage de Gibbs [Lawrence *et al.*, 1993]) de poser à nouveau dans ses propres termes le problème que nous cherchons à résoudre tel que celui-ci a été énoncé dans [Lawrence and Reilly, 1990] :

Le problème de Lawrence *Soit un ensemble de N chaînes dont on sait que toutes contiennent un signal biologique de longueur k. Le problème est d'identifier la localisation de ce signal sur chacune des chaînes.*

Observons que dans son énoncé original, la longueur du signal est connue, et que celui-ci est supposé être présent dans chacune des chaînes de l'ensemble. Ces hypothèses ne sont pas vraiment nécessaires mais travailler avec elles simplifie l'explication de l'algorithme, aussi nous allons les accepter pour l'instant.

Si nous reformulons le problème en termes plus familiers d'un bloc, ce qu'on nous demande de faire est localiser l'emplacement du bloc de longueur k et quorum N correspondant au signal porté par un ensemble de N chaînes, c'est-à-dire repérer son occurrence sur chacune de ces chaînes. Bien sûr la réussite d'une telle entreprise suppose que le bloc présente une certaine caractéristique permettant de le différencier de tous les autres blocs que l'on peut construire à partir de l'ensemble des N chaînes.

Cette caractéristique est justement celle qui correspond à la définition 3.4.11 donnée dans le chapitre 3. Le bloc recherché est ainsi celui qui maximize la valeur de :

$$\log L = \sum_{i=1}^{k} \sum_{j \in \Sigma} n_{ij} \log \frac{q_{ij}}{p_j}$$

où Σ est l'alphabet des monomères (nucléotides ou acides aminés), n_{ij} est le nombre de fois où le monomère j apparaît dans la colonne i de l'alignement sans trous des mots, q_{ij} est la fréquence d'apparition du monomère j à cette position et p_j est la fréquence d'apparition du monomère j sur les chaînes ailleurs que dans les mots du bloc. Rappelons que q_{ij} et p_j sont

éventuellement corrigés par des régularisateurs. Comment procède l'algorithme pour trouver ce bloc? La façon exacte, et naïve de faire consisterait à calculer la valeur de L pour tous les blocs possibles de quorum N et longueur k et voir lequel maximize cette valeur. Une telle approche serait en $O(n^N)$. L'heuristique de Lawrence consiste à choisir initialement un emplacement quelconque pour chaque mot du bloc correspondant au signal recherché, notons ce bloc B, sur chacune des chaînes de l'ensemble. L'algorithme procède alors en alternant une étape prédictive avec une étape d'échantillonnage de la façon suivante :

- étape prédictive : une des chaînes est sélectionnée, soit au hasard, soit l'une après l'autre dans un ordre fixé à l'avance. Notons cette chaîne s. Les descripteurs n_{ij}, q_{ij} et p_j sont calculés (éventuellement corrigés) d'après les positions actuelles des mots du bloc B sur toutes les chaînes à l'exception de s (cela signifie que le 'contenu' du bloc est figé par rapport à toutes les chaînes sauf s);

- étape d'échantillonnage : tous les mots possibles de longueur k sur la chaîne s sont alors considérés à tour de rôle comme candidats pour venir s'intégrer au bloc B et la valeur de la formule ci-dessus est établie par rapport à chacun de ces mots x sur s (rappelons que les mots sur les autres chaînes demeurent les mêmes). Notons cette valeur L_x. Un de ces mots est alors choisi avec une probabilité $\frac{L_x}{\sum_x L_x}$ pour venir se joindre au bloc B.

Les deux étapes sont répétées jusqu'à convergence, c'est-à-dire, jusqu'à ce que les positions des mots du bloc sur chacune des chaînes ne bouge plus d'une itération à l'autre, ou jusqu'à ce que la différence entre deux valeurs consécutives pour la formule devienne suffisamment petite pour que l'on considère l'apport d'information à partir de ce point statistiquement négligeable. L'idée derrière cet algorithme est que l'identification de l'emplacement correct du signal même sur une seule des chaînes de départ augmente le pouvoir qu'ont les descripteurs n_{ij}, q_{ij} et p_j de repérer ce signal sur les autres chaînes, et l'augmente d'autant plus (i.e. les descripteurs deviennent d'autant plus discriminants) que l'algorithme a progressé dans son déroulement. On peut comparer ce processus à celui de certaines formes d'apprentissages qui commencent lentement puis s'accélèrent à partir du moment où une certaine masse de connaissances a pu être accumulée de telle sorte que ce qui est appris 'à la fin' est assimilé beaucoup plus rapidement que ce qui doit être compris au départ. L'image d'un apprentissage dans ce cas précis est également appropriée dans la mesure où le choix d'un emplacement sur une des chaînes n'est jamais accompli de manière définitive. Ainsi un mauvais choix effectué à une étape donnée peut se trouver corrigé à une étape ultérieure lorsque le modèle (le bloc B) est devenu plus 'proche' du signal recherché. La convergence est assurée néanmoins car les choix ne sont pas non plus réalisés complètement au hasard. Observons que cela n'est pas vrai pour la version de 1990 ('Expectation Minimisation'). Pour cette version en effet, l'emplacement optimal du bloc sur chacune des chaînes n'est choisi qu'à la fin. L'étape d'échantillonnage est ainsi remplacée par une étape d'estimation qui calcule la matrice des fréquences de chaque monomère j, pour chaque position du bloc, lorsque celui-ci est placé sur tous les mots x de la chaîne, pondérées par la valeur L_x. Le 'bon' fonctionnement de l'algorithme dépend alors fortement du choix de la matrice initiale qui ne peut plus être réalisé de manière tout à fait aléatoire si l'on veut éviter les optimaux locaux. Nous verrons une illustration de cela dans le chapitre 6. La méthode d'échantillonnage de Gibbs a été élaborée précisément pour résoudre ce problème.

Ainsi que nous l'avons dit, ni la longueur du bloc (i.e. la longueur des mots le composant) ni son quorum n'ont besoin d'être fixés à l'avance. L'algorithme est capable également de rechercher plusieurs blocs simultanément, à condition toutefois que ceux-ci ne présentent pas de conflit entre eux. Lawrence ne peut rechercher que les trains de blocs, pas les blocs dont

les occurrences se croiseraient sur les chaînes. Une des façons possibles de procéder consiste à incorporer l'espacement entre les blocs directement dans la formule caractérisant la quantité d'information que ces blocs représentent [Cardon and Stormo, 1992]. L'algorithme de Lawrence a une complexité en temps qui est $O(T.n.k.N)$ où k est la longueur du bloc et T est un facteur qui est proportionnel à n mais, en pratique, inversément proportionnel à N (l"apprentissage' se fait d'autant plus rapidement que le nombre de chaînes — i.e. le nombre d'exemples — est grand). La complexité en espace est $O(n.N)$.

Les algorithmes de recherche de mots communs par optimisation stochastique sont très efficaces en général, du point de vue aussi bien de leur temps d'exécution que des résultats qu'ils produisent. Ils représentent également une approche qui est surprenante sous bien des aspects. Nous pouvons illustrer la méthode de Lawrence en particulier par l'image de quelqu'un qui déterminerait l'emplacement d'un objet, un bloc de mots, dans un certain espace, un ensemble de chaînes, en jouant aux dés et en se servant pour cela de dés qui auraient été convenablement 'pippés'. C'est une méthode tout à fait valable (et validée par la qualité des résultats obtenus), et de plus fondée sur une théorie mathématiquement justifiable. Bien sûr, la théorie s'appuie sur un modèle statistique qui simplifie sans doute la réalité biologique, mais qui est admissible en première approximation. Ce qui peut paraître plus troublant par contre, c'est que si cette approche permet de détecter des signaux, elle ne donne aucune information sur les mécanismes potentiels associés à ces signaux. Ceci vient du fait qu'elle se base sur une fonction du contenu global du signal et non sur la structure fine de celui-ci.

4.2.3 Méthodes combinatoires exactes

4.2.3.1 Introduction

Afin de compléter ce chapitre consacré aux méthodes directes de la ressemblance qui est aussi un chapitre de présentation du travail accompli par d'autres que nous, nous devrions maintenant décrire toutes les approches combinatoires exactes qui permettent d'identifier des blocs de mots communs, identiques ou similaires, dans un ensemble de chaînes par une comparaison de ces mots entre eux. Ces approches utilisent la relation de similarité entre monomères donnée dans la définition 3.4.10 du second chapitre et peuvent être divisées en deux grandes catégories. La première englobe toutes les méthodes travaillant avec une relation R de similarité qui est en fait l'identité ou une relation d'équivalence, et la seconde pouvant employer une relation symétrique, réflexive mais non transitive quelconque. Dans le premier cas, les blocs de mots communs trouvés forment des classes de mots identiques ou équivalents, dans le second, ils forment des cliques maximales de la relation entre monomères étendue aux mots. En termes d'algorithmes, les méthodes combinatoires directes et exactes peuvent également être divisées en trois autres groupes différents des précédents, le premier utilisant des techniques simples d'indexation pour localiser ces blocs de similarité, la seconde se servant de structures particulières de données comme les arbres ou les automates des suffixes (ces deux premiers groupes sont entièrement inclus dans celui des méthodes pour lesquelles R est l'identité ou une relation d'équivalence), et enfin le troisième employant des techniques d'attribution d'étiquettes qui sont aussi des techniques de partitionnement ou de raffinement d'ensembles pouvant être réalisées sans stocker autant d'information que les deux premières approches.

Comme ces dernières sont à la base des méthodes indirectes de comparaison de chaînes que nous avons élaborées dans le cadre de ce travail, nous allons donc nous contenter ici d'en fournir l'idée essentielle sous un aspect seulement — celui d'une attribution d'étiquettes — en laissant pour le chapitre 5 une discussion plus approfondie de leurs autres caractéristiques. Avant cela nous présentons les algorithmes de recherche de mots identiques par indexation

simple ou utilisant une arbre ou un automate des suffixes.

4.2.3.2 Recherche de mots communs identiques par indexation simple

Commençons par revenir un peu en arrière pour rappeler l'algorithme de MACAW que nous avons vu précédemment et qui permet de repérer des blocs de mots similaires sans trous dans un ensemble de chaînes. Supposons que les blocs recherchés par MACAW aient été en fait des blocs de mots identiques (il suffit pour cela de fixer le score de substitution d'un monomère par lui-même à une valeur supérieure au seuil t et par un autre monomère à -∞). MACAW aurait besoin de réaliser $O(n^2.N^2)$ opérations pour trouver ces blocs. Cela paraît excessif.

Considérons maintenant des algorithmes comme ceux de Leung [Karlin *et al.*, 1988a] [Karlin *et al.*, 1988b] [Leung *et al.*, 1991], de Martinez [Martinez, 1983] [Martinez, 1988] [Sobel and Martinez, 1986] ou de Santibanez [Santibanez and Rohde, 1987] qui identifient les blocs de mots identiques en codant ces mots sur des valeurs entières qui leur servent d'indexes. Ces mots indexés sont soit placés dans une table [Karlin *et al.*, 1988a] [Karlin *et al.*, 1988b] [Leung *et al.*, 1991] ce qui permet de les classer en un temps $O(n.N)$, soit triés dans des listes [Martinez, 1983] [Martinez, 1988] [Santibanez and Rohde, 1987] [Sobel and Martinez, 1986] qui peuvent être construites et manipulées en un temps $O(n.N.\log(n.N))$. Dans le premier cas, la longueur des mots communs repérables initialement est limitée par la taille de la table et ne peut être prolongée qu'à travers l'usage de listes chaînées ce qui ramène la complexité à $O(n.N.\log(n.N))$ comme pour les algorithmes de Martinez et Santibanez, mais est moindre cependant que celle de MACAW. Bien sûr, MACAW peut accepter des substitutions dans les blocs sans augmenter sa complexité algorithmique. Les trois algorithmes mentionnés ici permettent aussi des substitutions, dans une bien moindre mesure cependant, puisque la relation qui joue entre les symboles (et donc les mots) n'a pas besoin d'être l'identité mais peut être simplement une relation d'équivalence. L'algorithme de Leung admet également des substitutions dans un sens plus général, ainsi que des trous, mais uniquement entre les blocs de mots identiques identifiés au préalable et à condition que les régions d'erreurs ne dépassent pas une certaine longueur. La caractérisation du résultat final obtenu est alors impossible à formuler de manière précise aussi l'algorithme avec ces régions d'erreurs ne peut plus être qualifié d'exact. En fait, nous pouvons voir ces blocs de mots identiques initialement repérés par Leung, ainsi que ceux trouvés par Martinez et Santibanez, comme des points d'ancrage pour un alignement multiple *a posteriori* des chaînes de départ de façon similaire à ce qui est fait dans MACAW. Les critères utilisés pour sélectionner et éventuellement classer les blocs sont toutefois différents en ce qui concerne les algorithmes de Martinez et Santibanez puisque dans leur cas ces critères ont pour base des scores et non une évaluation statistique.

Observons que la classification des algorithmes de cette section parmi les méthodes exactes est un peu contestable. Elle n'est correcte que si l'on s'arrête pour chacun d'eux à la construction des blocs de mots communs identiques ou équivalents. Les prolongements de ces blocs de manière inexacte par Leung ou l'utilisation des blocs comme points fixes entre lesquels aligner des régions moins conservées [Martinez, 1983] [Martinez, 1988] [Santibanez and Rohde, 1987] [Sobel and Martinez, 1986] les situe parmi les heuristiques soit locales, soit globales respectivement.

4.2.3.3 Recherche de mots communs identiques utilisant un arbre ou un automate des suffixes

Bien que n'ayant pas été initialement conçus dans ce but, les arbres des suffixes [Chen and Seiferas, 1985] [McCreight, 1976] [Ukkonen, 1992] [Ukkonen and Wood, 1993] [Weiner, 1973] et les

automates des suffixes appelés DAWGs ('Directed Acyclic Word Graph') [Blumer *et al.*, 1985] [Blumer *et al.*, 1983] [Crochemore, 1985] [Crochemore, 1986] peuvent être utilisés pour trouver soit le plus long mot (facteur) commun à un ensemble de chaînes, soit tous les mots communs de longueur 2 à la longueur maximale où par mots communs on entend des mots identiques (ou équivalents moyennant une relation d'équivalence sur l'alphabet et une réécriture des chaînes). Une telle utilisation a déjà été effectuée dans le cadre soit de la recherche de mots répétés de manière contiguë dans une seule chaîne (ce sont les mots carrés ou qui sont puissances d'un même mot primitif — voir en particulier Apostolico [Apostolico and Preparata, 1983] et [Kosaraju, 1994] qui travaillent tous deux avec un arbre des suffixes), soit de la recherche de mots répétés non nécessairement contigus dans une ou plusieurs chaînes (Gusfield [Gusfield, 1995] et Hui [Hui, 1992] qui se servent d'un arbre des suffixes pour cela, Clift qui emploie un DAWG [Clift *et al.*, 1986]).

Les arbres aussi bien que les automates des suffixes réalisent en fait une indexation d'une ou de plusieurs chaînes qui est plus compacte que les indexes établis avec des tables ou des listes chaînées, plus efficaces à utiliser dans la plupart des applications, et permettent de révèler la structure interne des chaînes. Il n'est pas notre propos de détailler ici leur mode de construction, les personnes intéressées peuvent consulter les articles mentionnés plus haut ou le livre de M. Crochemore [Crochemore and Rytter, 1994], mais seulement de donner une idée très succinte de ce que représentent ces deux structures et comment les utiliser pour résoudre le problème qui nous concerne : localiser les blocs de mots communs (ici identiques) d'un ensemble de chaînes.

Considérons d'abord le cas le plus simple d'un arbre des suffixes pour une seule chaîne $s = s_1...s_n$. Un arbre des suffixes A pour s est un arbre orienté ayant une racine r et exactement n feuilles numérotées de 1 à n. Chaque nœud interne de l'arbre autre que r possède au moins deux fils et chaque branche de l'arbre est étiquetée par un mot non vide de s sachant que deux branches issues d'un même nœud ne peuvent recevoir des étiquettes commençant avec le même symbole. La caractéristique principale de l'arbre A est que, pour toute feuille i, la concaténation des étiquettes des branches formant le chemin de r à i épelle exactement le suffixe $s_i...s_n$ de s. Un exemple de l'arbre des suffixes pour la chaîne $s = aabbabd$ est donnée dans la figure 4.20. Bien sûr, pour des raisons d'espace, les branches ne sont pas vraiment étiquetées par des mots mais plutôt par une paire d'entiers indiquant les positions de début et fin du mot dans s (voir la figure 4.21).

La définition d'un arbre des suffixes telle que donnée ci-dessus ne garantit pas que celui-ci existe vraiment pour toute chaîne s. En effet, si un des suffixes de s est un préfixe d'un autre suffixe de s (par exemple, la chaîne $s' = aabbab$ possède un suffixe ab qui est préfixe du suffixe $abbab$ de s'), alors aucun arbre des suffixes correspondant à la définition ne serait possible puisque le chemin dans l'arbre relatif au premier suffixe n'aboutirait pas à une feuille. Pour éviter ce problème, nous devons donc supposer que le dernier symbole de s apparaît nulle part ailleurs dans s. Il suffit pour cela d'ajouter un symbole spécial à la fin de la chaîne s qui n'appartienne pas à l'alphabet sur lequel s est défini. Un arbre des suffixes pour une chaîne s de longueur n peut être construit en un temps $O(n)$ et occupe un espace également $O(n)$ [McCreight, 1976].

Étant donné un tel arbre pour une chaîne s, les mots répétés de s correspondent alors aux étiquettes concaténées des chemins dans l'arbre qui commencent à la racine et se terminent dans un des nœuds internes (voir la figure 4.23). Ils peuvent être énumérés par un simple parcours en profondeur dans l'arbre. Le quorum des blocs qu'ils forment est donné par le nombre de feuilles du sous-arbre ayant le nœud interne pour racine.

L'arbre des suffixes de $N \geq 2$ chaînes est quant à lui construit de manière très similaire à l'arbre pour une chaîne, il suffit simplement de distinguer les chaînes entre elles en concaténant

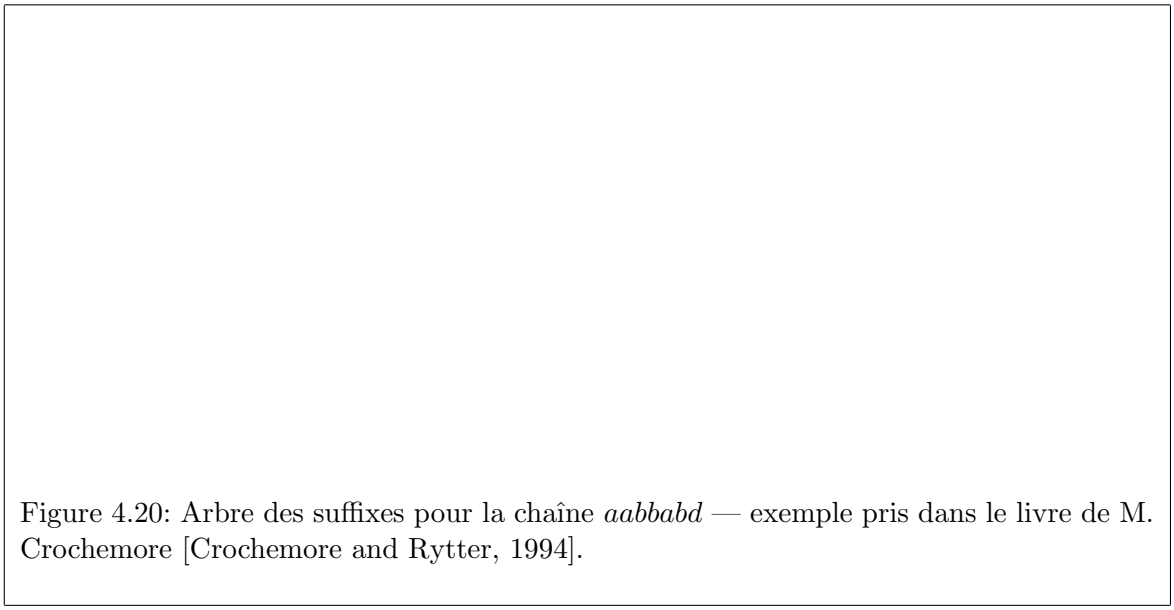


Figure 4.20: Arbre des suffixes pour la chaîne *aabbabd* — exemple pris dans le livre de M. Crochemore [Crochemore and Rytter, 1994].

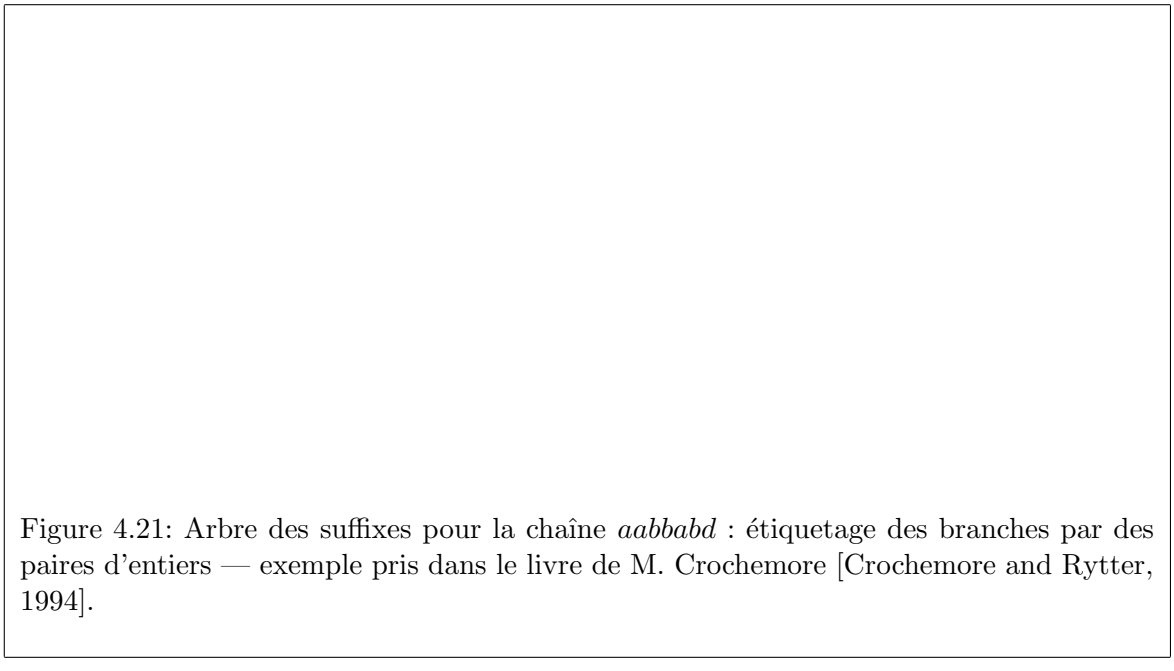


Figure 4.21: Arbre des suffixes pour la chaîne *aabbabd* : étiquetage des branches par des paires d'entiers — exemple pris dans le livre de M. Crochemore [Crochemore and Rytter, 1994].

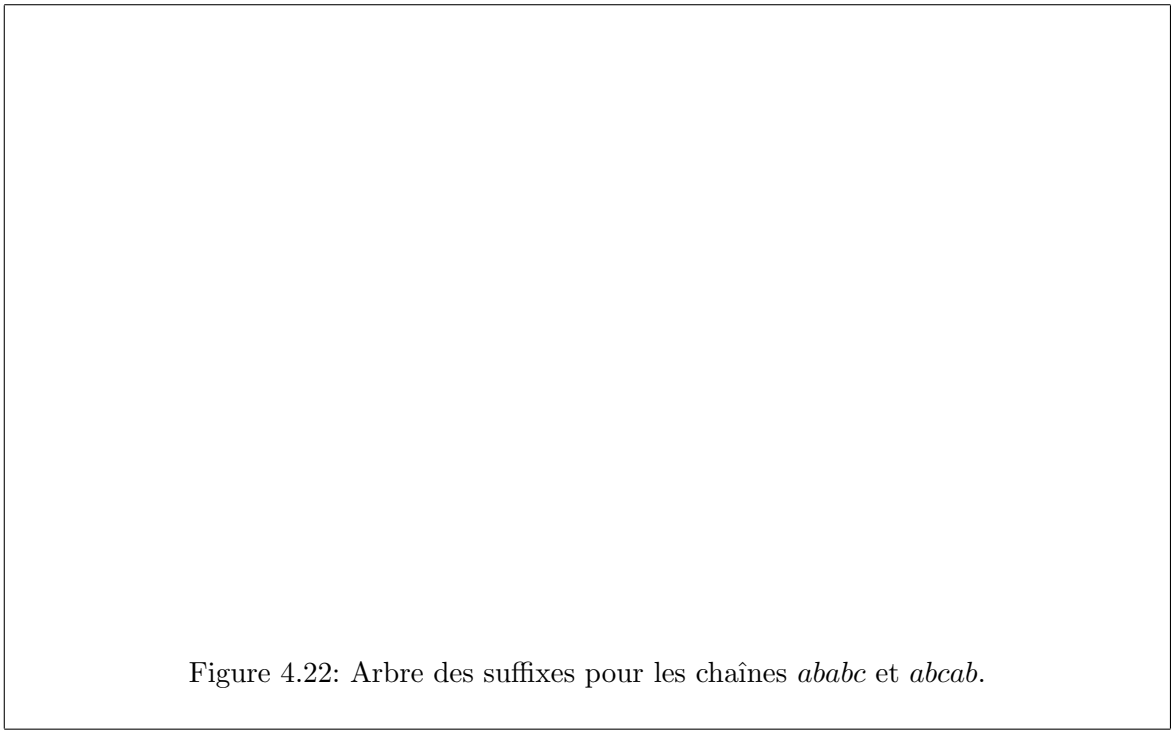


Figure 4.22: Arbre des suffixes pour les chaînes *ababc* et *abcab*.

à la fin de chacune un symbole spécial non présent dans l'alphabet et distinct des symboles spéciaux accolés aux autres chaînes (voir la figure 4.22). Cet arbre possède $n.N$ feuilles si l'on considère des chaînes de même longueur et sont numérotées de $(1,1)$ à (N,n). Étant donné une feuille (p,i), la concaténation des étiquettes des branches formant le chemin de la racine r à (p,i) épelle cette fois le suffixe $sp_i...sp_n$ de la chaîne sp de l'ensemble. Gusfield propose alors un algorithme qui utilise l'arbre des suffixes de $N \geq 2$ chaînes pour trouver le plus long bloc de mots identiques satisfaisant la contrainte de quorum q pour q allant de 2 à N en un temps $O(q.n.N)$, réduit ensuite à $O(n.N)$ par Hui [Hui, 1992]. Notons que le quorum dans le cas de N chaînes correspond bien sûr au nombre de feuilles se rapportant à des chaînes distinctes.

La différence fondamentale entre un arbre des suffixes et un DAWG est que le DAWG est obtenu en identifiant les sous-arbres isomorphiques de l'arbre non compacté A' représentant les suffixes d'une (ou de plusieurs) chaîne s. A' est l'arbre A donné précédemment (figure 4.20), où chaque branche étiquetée par un mot de longueur l supérieure à 1 est remplacée par l branches étiquetées par un unique symbole de l'alphabet (voir la figure 4.24). Certains nœuds internes de l'arbre A' peuvent n'avoir qu'un seul fils. Le DAWG correspondant est donné dans la figure 4.25, il constitue ainsi une forme compactée de l'arbre A' différente de celle obtenue par l'arbre des suffixes. Dans le cas d'une seule chaîne, chaque nœud du DAWG représente la classe d'équivalence de tous les mots ayant mêmes positions finales dans s. Ces positions et/ou leur nombre peuvent être aisément stockés dans les nœuds du DAWG lors de la construction de ce dernier et les mots répétés dans s correspondent alors aux étiquettes concaténées de tous les chemins allant de la source jusqu'à un nœud pour lequel ce nombre est supérieur à 1. Un avantage du DAWG par rapport aux arbres des suffixes est que toute arête de l'automate est étiquetée par un seul symbole. C'est donc une structure plus pratique à utiliser lorsque l'information qui nous intéresse est celle associée aux arêtes plutôt qu'aux nœuds.

La construction d'un DAWG demande le même temps que celle d'un arbre des suffixes et

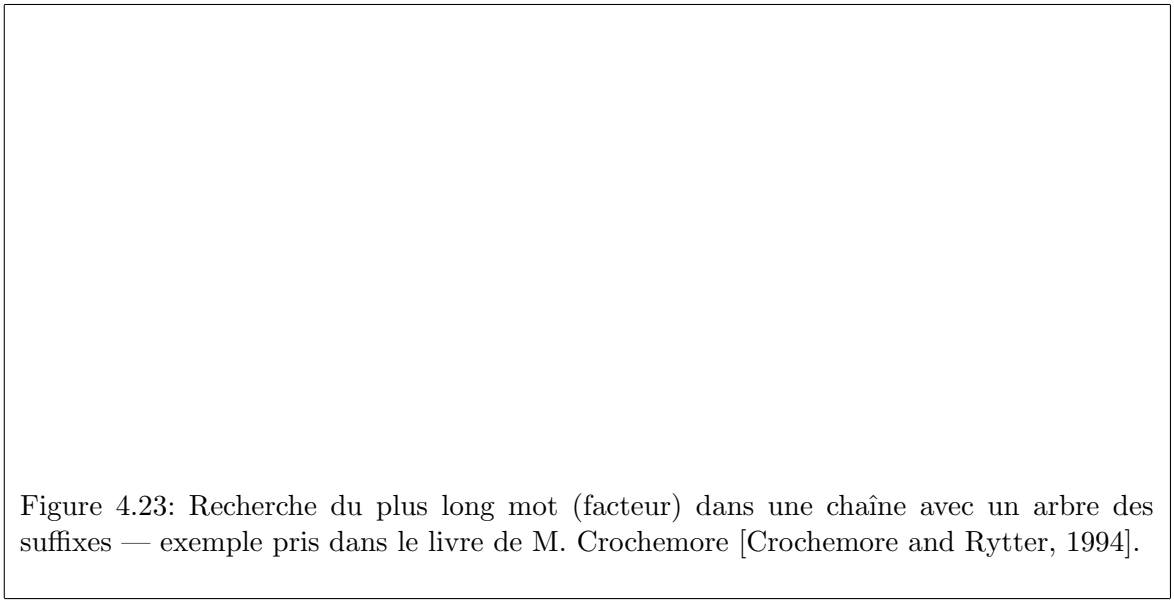


Figure 4.23: Recherche du plus long mot (facteur) dans une chaîne avec un arbre des suffixes — exemple pris dans le livre de M. Crochemore [Crochemore and Rytter, 1994].

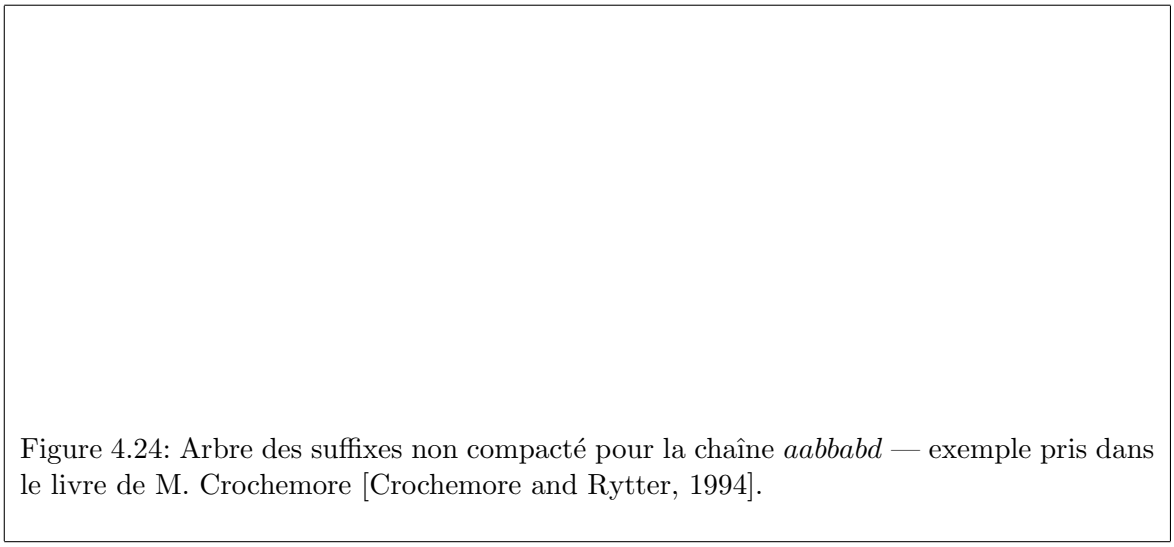


Figure 4.24: Arbre des suffixes non compacté pour la chaîne *aabbabd* — exemple pris dans le livre de M. Crochemore [Crochemore and Rytter, 1994].

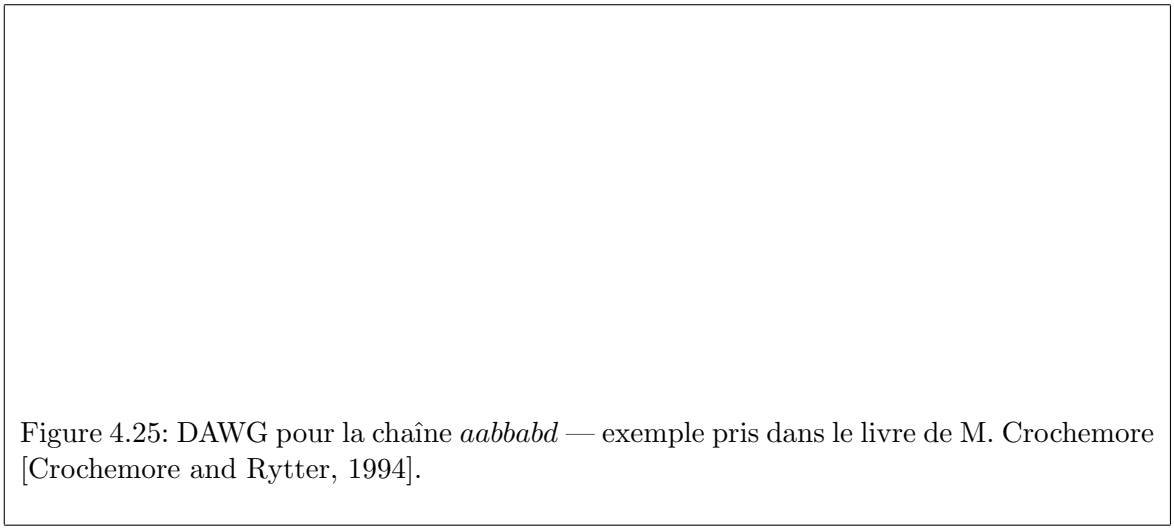


Figure 4.25: DAWG pour la chaîne *aabbabd* — exemple pris dans le livre de M. Crochemore [Crochemore and Rytter, 1994].

l'espace occupé est également proportionnel à la longueur de la chaîne (ou de toutes les chaînes s'il y en a plus d'une). De même que l'arbre des suffixes, le DAWG a été employé par Clift *et al.* [Clift *et al.*, 1986] notamment pour trouver tous les mots répétés dans une séquence d'ADN dans le cadre spécifique cette fois d'une application en biologie.

4.2.3.4 Recherche de mots communs identiques ou similaires par attribution d'étiquettes

Considérons le cas d'une seule chaîne s, et la recherche de mots répétés de manière identique dans s. Que pouvons-nous dire de la composition d'un mot de longueur $k > 1$ — notons-le u — qui se trouve présent au moins deux fois dans s à des positions différentes, par exemple i et j? Notons v et w les deux mots non vides de longueurs k' et k'' respectivement tels que $u = vw$. Nous pouvons alors affirmer que v et w sont aussi des mots répétés au moins deux fois dans s, en particulier v est présent aux positions i et j et w aux positions $i + k'$ et $j + k'$. Tout mot répété de longueur $k > 1$ est ainsi toujours composé de deux mots répétés juxtaposés dans s de longueurs $k' \geq 1$ et $k'' \geq 1$ telles que $k' + k'' = k$.

Cette idée simple forme la base d'un algorithme élaboré par Karp, Miller et Rosenberg en 1972 [Karp *et al.*, 1972] pour trouver tous les mots répétés de façon identique dans une chaîne et que nous appelerons dorénavant KMR. Elle est formellement présentée dans le lemme donné ci-dessous et illustrée dans la figure 4.26. E représente une relation d'équivalence sur Σ qui, dans le cas de KMR, est l'identité I.

Lemme 4.2.1 *Soient $s \in \Sigma^n$, $a, b \in \{1, ..., n\}$ et $i, j \in \{1, ..., n - (a + b) + 1\}$. Alors :*

$$i \; E_{a+b} \; j \text{ si et seulement si } i \; E_a \; j \text{ et } i + a \; E_b \; j + a.$$

La décomposition d'un mot u peut en fait s'effectuer de plusieurs façons. Ainsi, soient z et t deux mots non vides distincts de v et w et tels que $u = vw = zt$, et supposons sans perte de généralité que $| \; v \; | > | \; z \; |$: nous pouvons alors affirmer que u est un mot présent au moins deux fois dans s si et seulement si v et t sont eux-mêmes des mots répétés. Cette observation forme le second lemme dont KMR se sert pour trouver tous les mots répétés dans une chaîne s. Ce lemme est donné ci-dessous et illustré dans la figure 4.27.

Figure 4.26: Une illustration du premier lemme de KMR.

Figure 4.27: Une illustration du second lemme de KMR.

Lemme 4.2.2 *Soient $s \in \Sigma^n$, $a, b \in \{1, ..., n\}$ avec $b \leq a$ et $i, j \in \{1, ..., n-(a+b)+1\}$. Nous avons alors :*

$$i \; E_{a+b} \; j \text{ si et seulement si } i \; E_a \; j \text{ et } i+b \; E_a \; j+b.$$

Notons que, lorsque a est égal à b, les deux lemmes sont bien sûr identiques.

Une fois ces lemmes établis qui nous fournissent un moyen de repérer les mots répétés par récurrence, il y a plusieurs manières possibles de se représenter le fonctionnement actuel de l'algorithme. Nous pouvons le voir comme un algorithme de partitionnement d'ensembles (les classes d'équivalence de la relation d'identité entre symboles étendue aux mots), ou comme un parcours dans un arbre qui est étroitement lié à l'arbre des suffixes vu dans la section précédente (ce parcours est imaginaire dans la mesure où aucune structure n'est effectivement construite). Nous pouvons le voir également comme une technique d'attribution d'étiquettes. Il est intéressant d'explorer la question sous tous ces aspects, et nous le ferons en détail en ce qui concerne les deux premiers dans le prochain chapitre. Nous considérons ici uniquement le côté recherche de mots répétés comme une attribution d'étiquettes (numériques). Nous montrons que l'algorithme se réduit alors à un problème de tri. La présentation que nous en faisons est celle donnée par Crochemore [Crochemore and Rytter, 1994].

Considérons les mots répétés de longueur $k \geq 1$ d'une chaîne s et supposons qu'il existe q mots distincts dans s ayant cette longueur. Soit mot(1), mot(2),..., mot(q) la séquence de ces q mots en ordre lexicographique. Étant donné une position i dans s, nous établissons alors que :

$$\text{étiq}_k(i) = j \iff s_i ... s_{i+k-1} = \text{mot}(j).$$

La définition de ressemblance (ici l'identité) donnée dans le chapitre 3 (définition 3.4.10) devient :

$$i \; E_k \; j \iff \text{étiq}_k(i) = \text{étiq}_k(j).$$

L'algorithme de KMR consiste alors simplement en un calcul du tableau étiq_k pour tout k.

D'une façon générale, ce calcul se fait par application des lemmes. Supposons étiq_a et étiq_b connus, c'est-à-dire, supposons connues les deux listes :

$$\mathcal{L}_a = (\acute{e}tiq_a(1), \acute{e}tiq_a(2), ..., \acute{e}tiq_a(n-a+1))$$

$$\mathcal{L}_b = (\acute{e}tiq_b(1), \acute{e}tiq_b(2), ..., \acute{e}tiq_b(n-b+1))$$

(dans le cas du lemme 4.2.2, il suffit seulement en fait de connaître la liste $\mathcal{L}_a$). Voyons maintenant comment calculer étiq_{a+b} et $\mathcal{L}_{a+b}$. Soit la liste $\mathcal{L}'_{a+b}$ des étiquettes possibles pour les mots de longueur $a+b$ de s donnée par :

$$\mathcal{L}'_{a+b} = (\acute{e}tiq_a(1)\acute{e}tiq_b(1+a), 1), (\acute{e}tiq_a(2)\acute{e}tiq_b(2+a), 2), ...,$$
$$(\acute{e}tiq_a(n-a-b+1)\acute{e}tiq_b(n-b+1), n-a-b+1))$$

ou bien par :

$$\mathcal{L}'_{a+b} = (\acute{e}tiq_a(1)\acute{e}tiq_a(1+b), 1), (\acute{e}tiq_a(2)\acute{e}tiq_a(2+b), 2), ...,$$
$$(\acute{e}tiq_a(n-a-b+1)\acute{e}tiq_a(n-a+1), n-a-b+1))$$

où chaque étiquette de mot est cette fois associée avec la position dans s où ce mot apparaît. Une fois cette liste triée en ordre lexicographique sur le premier champ et partitionnée en classes ayant ce premier champ identique, il suffit de renuméroter ces classes à partir de 1 pour obtenir les étiquettes étiq_{a+b} des classes des mots identiques de longueur $a+b$. La liste $\mathcal{L}_{a+b}$ donnée par :

$$\mathcal{L}_{a+b} = (\acute{e}tiq_{a+b}(1), \acute{e}tiq_{a+b}(2), ..., \acute{e}tiq_{a+b}(n-a-b+1))$$

est quant à elle obtenue en réordonnant selon son second champ cette fois (indiquant les positions dans s) la liste $\mathcal{L}'_{a+b}$ préalablement triée de manière lexicographique.

Le coût de ces opérations se réduit à celui d'un tri lexicographique sur au plus n éléments de l'intervalle entier (1,2,...,n) qui peut être réalisé en un temps $O(n)$ [Aho *et al.*, 1983], plus celui d'un second tri sur les positions qui peut être réalisé également en $O(n)$ à l'aide d'un tableau auxiliaire de taille n. Si nous faisons $a = b$, l'algorithme KMR requiert donc $O(n.\log k)$ opérations pour repérer les blocs de mots identiques d'une chaîne s quelle que soit la longueur k du bloc ainsi qu'illustré dans la figure 4.28. Observons que k est au maximum égal à $n-1$.

Dans le cas où la longueur k des mots communs recherchés n'est pas une puissance de 2, on utilise un des deux lemmes avec $a = b$ jusqu'à la plus grande puissance de 2 inférieure à la longueur désirée, puis on recourt au lemme 2 avec $a \neq b$ afin de calculer nom_k. La recherche des plus longs blocs est effectuée de manière similaire. Les noms des blocs sont attribués toujours en doublant la longueur de ceux-ci à chaque étape, mais, cette fois, jusqu'à la plus petite puissance de 2 qui est supérieure à la longueur k_{max} pour laquelle il existe encore au moins un bloc de mots communs possédant au minimum deux éléments. La recherche des blocs de longueur notée k_{max} procède ensuite par dichotomie comme indiqué dans la figure 4.28.

L'extension au cas de N chaînes $s1, s2, ..., sN$ avec une contrainte de quorum a été réalisée par Landraud [Landraud *et al.*, 1989] et est immédiate. Les N chaînes sont concaténées en une

/* Recherche de tous les blocs de mots répétés identiques de longueur k. */

construire le tableau étiq$_1$;
soit d la plus petite puissance de 2 inférieure à k ($d = 2^{\lfloor \log_2 k \rfloor}$)
appliquer le lemme 4.2.1 ou 4.2.2 avec $a = b$ pour construire les tableaux étiq$_2$, étiq$_4$, étiq$_8$,..., étiq$_d$;
si $k = d$
    arrêt;
sinon
    utiliser le lemme 4.2.2 pour construire étiq$_k$ à partir de étiq$_d$.

/* Recherche de tous les blocs des plus longs mots répétés identiques. */

construire le tableau étiq$_1$;
appliquer le lemme 4.2.1 ou 4.2.2 avec $a = b$ pour construire les tableaux d'étiquettes étiq$_2$, étiq$_4$, étiq$_8$,..., étiq$_d$ où d est la plus petite puissance de 2 pour laquelle les blocs obtenus possèdent encore au moins deux éléments;
réaliser une recherche dichotomique pour trouver les plus longs blocs :
    étiq$_{\frac{d}{2}}$ est utilisé pour construire étiq$_{\frac{3d}{4}}$ (lemme 4.2.2)
    si au moins une étiquette dans étiq$_{\frac{3d}{4}}$ correspond à un bloc possédant au minimum deux éléments
        utiliser étiq$_{\frac{3d}{4}}$ pour construire étiq$_{\frac{7d}{8}}$;
    sinon
        utiliser étiq$_{\frac{d}{2}}$ pour construire étiq$_{\frac{5d}{8}}$ etc.

Figure 4.28: Schéma de l'algorithme KMR.

seule — notons-la encore s — et les blocs de mots identiques sont repérés dans cette chaîne $s = s1s2...sN$ en prenant soin de ne pas permettre qu'un mot répété dans un bloc appartienne en fait à deux chaînes différentes. Il suffit pour cela de placer entre les chaînes $s1, s2, ..., sN$ dans s un symbole spécial non présent dans l'alphabet. Ainsi s est en réalité égal à $s1\$s2\$...\$sN$ avec $\$ \notin \Sigma$. En conservant les positions de début de chacune des chaînes originales dans s, la vérification de la contrainte de quorum demande un simple parcours du tableau des noms et se réalise en un temps $O(n.N)$. La complexité en temps finale de l'algorithme est en $O(n.N.\log k)$ où k est au maximum égal à n. L'espace requis est $O(n.N)$.

La substitution d'une relation d'identité ou d'équivalence par une relation quelconque non transitive R ne change presque rien au fonctionnement de l'algorithme proprement dit. Les deux lemmes demeurent valables, E est simplement remplacé par R.

Les blocs repérés ne représentent toutefois plus des classes de mots identiques ou équivalents, mais des cliques de mots tous en relation deux à deux par R_k. Comme R est non transitive, un mot de longueur k peut également appartenir à plus d'une clique et avoir ainsi plus d'une étiquette. En fait, un mot possède entre un et g^k étiquettes, où g mesure le degré de non-transitivité de R (voir la section 3.2.3.2 du chapitre 3). La liste $\mathcal{L}'$ des étiquettes de longueur $k = (a + b)$ donnée précédemment pour $R = E$ devient ainsi une liste plus étendue, où une même position peut apparaître jusqu'à g^k fois comme second champ d'un élément. Le tri lexicographique de $\mathcal{L}'$ se fait de la même façon, ainsi que la séparation de la liste triée en groupes ayant un premier champ identique. Cette séparation demeure un partitionnement par rapport à ce premier champ, mais ne l'est plus par rapport au second dénotant les positions des mots dans s. Il suffit ensuite de numéroter ces groupes à partir de 1 pour obtenir le tableau étiq_k des cliques de longueur k.

Le processus de construction des tableaux étiq_k pour un k quelconque, ou du tableau $\text{étiq}_{k_{max}}$, et donc des cliques est semblable à celui adopté par KMR et indiqué dans l'algorithme de la figure 4.28. La longueur des mots similaires repérés est ainsi doublée à chaque étape tant que cela est possible. La complexité en temps de cette opération pour le cas de $N \geq 1$ chaînes de longueur moyenne n est $O(n.N.g^k.\log k)$ et celle en espace est $O(n.N.g^k)$.

Les cliques obtenues ne sont bien sûr pas nécessairement maximales. La chaîne est en effet de longueur finie et certains symboles appartenant à la différence de deux cliques maximales de R peuvent ne jamais y être présents. Si l'on souhaite travailler avec des blocs de mots qui représentent de telles cliques maximales, il faut encore, après le tri et la séparation de la liste $\mathcal{L}'$ suivant la valeur du premier champ, effectuer des tests d'inclusion des ensembles de positions obtenus (correspondant aux seconds champs des éléments de $\mathcal{L}'$) afin d'éliminer les cliques incluses dans une autre. Pour chaque étape k, cette opération peut se faire en $O(n.N.k.g^{2k})$ opérations [Soldano *et al.*, 1995], toujours dans le cas de $N \geq 1$ chaînes. Les étiquettes correspondant à des cliques non-maximales sont exclues de $\mathcal{L}'$ et la numérotation procède alors uniquement sur les cliques maximales restantes. L'algorithme complet s'appelle KMR Cliques abrégé en KMRC [Soldano *et al.*, 1995].

Nous reviendrons sur KMR et KMRC dans le chapitre 5, vus cette fois comme des cas spéciaux des méthodes de comparaisons indirectes de chaînes. Ces algorithmes seront alors abordés sous deux angles un peu différents de celui présenté ici mais qui y apparaissent déjà en filigrane : l'angle partitionnement/raffinement d'ensembles (de positions) et l'angle parcours dans un arbre. Notons pour clore cette section que, comme dans le cas de MACAW ou des algorithmes de la section 4.2.3.2, les blocs de mots communs identiques ou similaires obtenus par KMR ou KMRC peuvent être utilisés comme points d'ancrage pour un alignement multiple des chaînes. Un positionnement optimal des blocs est un problème NP-complet et seules des heuristiques ont été proposées à ce jour [Landraud *et al.*, 1989] [Viari and Pothier, 1993].

4.2.3.5 Le cas des structures

Ainsi que nous l'avons vu dans la section 3.2.3.2.2, il est possible d'établir une relation entre monomères en tant qu'objets 3D cette fois. Cette relation possède les mêmes caractéristiques que celle portant sur les monomères en tant que symboless, simplement elle intervient sur un alphabet plus grand et est définie de manière différente, sur une notion de voisinage de carrés dans un plan correspondant à une équivalence entre paires d'angles (les coordonnées internes de la macromolécule). Une version de KMRC adaptée à la comparaison multiple de structures de protéines codées en chaînes de caractères a ainsi été réalisée [Viari, 1995]. Une innovation a cependant été effectuée dans l'algorithme, qu'il est intéressant de noter au passage car elle introduit l'idée d'un 'mot spécial', composé de symboles non contigus.

L'introduction de cette innovation a été nécessaire à cause d'une particularité des objets à comparer (la structure des protéines) qui rendait parfois l'approche 'normale' de KMRC inefficace en pratique. KMRC en effet traite les structures comme des chaînes et donc les motifs structuraux comme des suites contiguës de caractères. Or la structure d'une protéine possède, ainsi que nous l'avons vu dans le chapitre 2, un certain nombre de régularités locales associées à ce qui est appelé sa structure secondaire. En particulier, elle est composée d'un nombre plus ou moins grand d'hélices. En termes de coordonnées internes, ces dernières sont représentées par des suites contiguës de paires d'angles approximativement identiques, et sont donc codées par des symboles identiques dans le nouvel alphabet obtenu par échantillonnage de la carte de Ramachandran (voir la section 3.2.2.3). Les chaînes de caractères correspondant à ces structures peuvent ainsi présenter de nombreux mots (de longueur entre 4 et 20 en général) composés de la répétition d'un même symbole. Chercher les mots communs à un ensemble de telles chaînes en les repérant par longueurs croissantes à la manière KMR ou KMRC peut ainsi consommer beaucoup de temps et d'espace en mémoire puisque typiquement ces chaînes peuvent avoir un très grand nombre de mots en relation par R_k pour des valeurs petites de k (autour de 4-5). Ces mots correspondent le plus souvent à de petites hélices (ou de petits morceaux d'hélices) et ne sont en général pas très intéressants. L'idéal était donc de réussir à les éviter, tout en ne perdant pas l'information nécessaire pour identifier tous les motifs structuraux plus longs (grandes hélices ou autres structures). L'idée a consisté à travailler, au moins au départ, non avec des 'vrais mots', c'est-à-dire, avec des mots composés de suites contiguës de symboles, mais avec des 'mots spéciaux'. Par exemple, nous pouvons définir un 'mot spécial' de longueur 2 comme étant un 'vrai mot' de longueur 8 dont seules les première et dernière positions 'comptent'. Deux 'mots spéciaux' de longueur 2 sont alors en relation par R_2 si et seulement si leurs premier et dernier symboles sont en relation par R. L'algorithme recherche ainsi les mots répétés en 'sautant' le long de la chaîne (d'où son nom de KMRS — S comme 'Saut' [Sagot *et al.*, 1995b]) jusqu'à ce qu'une certaine longueur k_0 soit atteinte à partir de laquelle les mots redeviennent 'normaux' et l'algorithme KMRS exactement pareil à KMRC. La complexité en temps finale de KMRS est en $O(n.N.k.(k_0 + \log(k - k_0).g^{2k})$, celle en espace demeure la même qu'indiqué ci-dessus.

4.2.3.6 Recherche de mots communs avec erreurs — une transition vers les méthodes indirectes

Mentionnons enfin un algorithme qui ne peut être classé ni parmi les méthodes de recherche de mots communs utilisant un arbre de suffixes, ni parmi celles de comparaison indirecte bien qu'il soit apparenté aux deux. L'algorithme de Lefevre [Lefevre and Ikeda, 1993a] [Lefevre and Ikeda, 1993b] [Lefevre and Ikeda, 1994] se sert en effet d'un arbre des suffixes (l'arbre des positions de Weiner, moins compact que celui de McCreight) pour repérer des blocs de mots similaires cette

fois et non plus uniquement identiques ou équivalents, ce qui veut dire qu'un certain nombre de substitutions (pas de trous) est permis entre les mots. L'arbre n'est cependant utilisé que pour produire un index exact de tous les mots dans les chaînes et les blocs sont ensuite formés simplement en se servant de l'arbre pour comparer tous ces mots entre eux. Chaque bloc est ainsi composé de tous les mots se trouvant à une certaine distance de Hamming d'un mot présent dans une des chaînes (dans les méthodes indirectes, les objets contre lesquels les mots dans les chaînes sont comparés sont externes à ces dernières et peuvent ne même pas être des mots — voir la section 3.3.3 du chapitre 3).

Le fait que des substitutions soient autorisées signifie que la comparaison d'un mot avec l'arbre donne lieu à plusieurs parcours dans celui-ci. Si la longueur des blocs identifiés est fixe, notons-la m, et si aucune substitution n'est permise, la complexité de l'algorithme est $O(n.N.m)$. Si l'on recherche les blocs similaires de longueur maximale, toutes les longueurs possibles de blocs (satisfaisant une contrainte de quorum allant de 2 à N) doivent être traitées au préalable et la complexité devient alors $O(n^3.N)$ dans le pire des cas, toujours sans erreurs. Enfin si e substitutions sont autorisées, la complexité en théorie est $O(e.n.N.m)$ ou $O(e.n^3.N)$ respectivement (étant donné un motif u de longueur m, il faut $O(e.m)$ opérations en utilisant un arbre des suffixes pour déterminer s'il existe un mot v d'un ensemble de chaînes tel que la distance de Hamming entre u et v est au plus de e [Gusfield, 1995]). La complexité en espace est toujours $O(n.N)$.

Chapitre 5

Les méthodes indirectes de comparaisons

Tell all the Truth but tell it slant -
Success in Circuit lies ...

Poème #1129. Emily Dickinson.
The Complete Poems of Emily Dickinson.
Thomas H. Johnson (Ed.). Faber and Faber, 1975.

Ce chapitre est consacré aux méthodes indirectes de comparaison de chaînes. À trois exceptions près qui seront clairement indiquées, tous les algorithmes introduits ont été réalisés dans le cadre de ce travail.

Nous commençons par rappeler le problème, et en particulier le fait que les comparaisons entre chaînes d'un ensemble sont faites désormais en utilisant des objets externes à ces chaînes appelés des modèles, et nous le posons en termes formels. Puis nous décrivons deux des trois algorithmes élaborés à ce jour par d'autres que nous pour essayer de le résoudre.

Nous réalisons ensuite une visite générale de l'approche que nous proposons nous-mêmes. Cette visite approfondit des idées déjà suggérées par les méthodes combinatoires exactes directes présentées à la fin du chapitre 4. Nous reprenons ainsi le principe d'une construction par récurrence des objets recherchés. Ce principe a été introduit dans le chapitre précédent sous la forme d'une attribution d'étiquettes, nous le revoyons ici comme un parcours dans un arbre et comme une technique de partitionnement ou de raffinement d'ensembles. L'algorithme prototype et une première analyse intuitive de sa complexité sont alors discutés.

Après cette visite générale, nous passons à une visite détaillée des méthodes que nous avons élaborées. Les ressemblances entre mots que ces méthodes permettent de dégager correspondent à celles définies dans la dernière section du chapitre 3. Les algorithmes proposés se divisent en deux grands groupes, celui pour lequel les symboles de l'alphabet constituent l'unité de comparaison entre mots dans les chaînes, l'autre pour lequel cette unité est elle-même composée de mots. Le premier groupe se subdivise encore en trois sous-groupes, le premier reflétant une idée biologique de la ressemblance ayant pour base une distance (nombre de mutations séparant deux objets), la seconde essayant de se libérer de tout point de vue sur la nature de ce qui est semblable afin d'optimiser, dans une certaine limite, sur l'alphabet des modèles, et enfin le dernier qui tente de conjuguer les deux approches.

Nous n'abordons pas ici le problème du traitement à appliquer aux objets trouvés (les modèles et leurs occurrences). Ce chapitre ne comporte non plus ni les critiques que nous

désirons porter nous-mêmes sur le travail accompli, ni une discussion de ce qu'il serait intéressant d'essayer de réaliser encore à partir des bases établies ici. Ces aspects sont importants mais ne seront discutés que dans les deux derniers chapitres.

Celui que nous présentons maintenant comporte des aspects techniques, il nous semble en effet utile de discuter non seulement les idées générales mais aussi les détails d'implémentation des divers algorithmes, ainsi que les variantes et contraintes qu'il est possible et intéressant d'y introduire. Un effort a cependant été fait dans l'organisation du chapitre pour qu'un lecteur non intéressé par ce niveau de précision puisse les éviter. L'essentiel des idées de ce chapitre se trouve ainsi regroupé dans la section consacrée à un aperçu général de nos méthodes et dans le tableau-résumé de la fin.

5.1 Le problème formellement posé

Commençons par rappeler le problème spécifique que nous essayons de résoudre. Il s'agit toujours bien sûr de comparer plusieurs chaînes simultanément entre elles afin de repérer des mots présents dans ces chaînes qui se 'ressemblent' suivant une certaine définition de similarité. Dans la seconde partie du chapitre précédent, nous avons présenté diverses méthodes permettant de réaliser ce repérage. D'une manière un peu grossière, nous pouvions y distinguer deux approches. La première avait pour caractéristique d'autoriser une recherche souple, elle était cependant soit exacte mais trop coûteuse en temps de calcul (programmation dynamique appliquée au cas local), soit rapide mais faisant appel à des heuristiques (par exemple MACAW — section 4.2.2.1.2.2). La seconde, composée des méthodes locales combinatoires (section 4.2.3), était quant à elle rapide et exacte mais relativement peu souple. Les mots repérés étaient soit des mots identiques ou équivalents, soit des mots similaires mais liés les uns aux autres par une relation simple.

Notre problème est donc d'introduire de plus en plus de flexibilité dans la notion de similarité entre mots tout en demeurant exhaustifs dans la recherche de ces mots et relativement efficaces.

Ce que nous voulons c'est trouver tous les groupes de mots similaires suivant une des quatre définitions de similarité données dans la section 3.5 du chapitre 3 (définitions 3.5.3, 3.5.8, 3.5.12 et 3.5.15). Pour réaliser cela, les mots ne sont plus comparés directement entre eux, mais le sont à des objets externes que nous avons appelé des modèles dont une description a été fournie dans la section 3.5. Les groupes de mots similaires le sont parce qu'il existe un tel modèle auquel tous 'ressemblent' d'une certaine façon.

Une définition formelle du problème général que nous souhaitons résoudre peut alors être la suivante :

Le problème de la construction des modèles *Étant donné un ensemble de N chaînes $s1, s2, ..., sN \in \Sigma^*$ et un entier q entre 2 et N (représentant une contrainte de quorum — définition 3.3.5), les problèmes que nous nous proposons de résoudre sont :*

Problème 1 *Trouver tous les modèles, ainsi que leurs ensembles d'occurrences, d'une longueur k qui sont présents (L, EL, ECL ou w-présents) dans au moins q des chaînes de l'ensemble;*

Problème 2 *Trouver la plus grande longueur k_{max} pour laquelle il existe au moins un modèle présent (à nouveau, L, EL, ECL ou w-présent) dans au moins q des chaînes de l'ensemble et résoudre le problème 1 pour $k = k_{max}$.*

La question est maintenant, comment apporter une solution à ce problème général?

Sous des formes simples, des objets très semblables à des modèles ont été utilisés par d'autres avant nous, et nous commençons par présenter leur approche. Nous appelons cette approche naïve car elle consiste à engendrer tous les modèles possibles et à les rechercher dans l'ensemble des chaînes à comparer.

5.2 Approche naïve : engendrer tous les modèles

5.2.1 Des modèles comme des mots : l'algorithme de Waterman

Dans une série d'articles publiés depuis 1984 [Galas *et al.*, 1985] [Waterman, 1984b] [Waterman *et al.*, 1984b] [Waterman, 1986] [Waterman, 1989] [Waterman, 1990], Waterman a introduit l'idée d'utiliser un objet externe contre lequel sont effectuées toutes les comparaisons des mots présents dans un ensemble de chaînes. À notre connaissance, il est le premier à avoir employé cette technique dans une approche purement algorithmique (des approches semblables ont été élaborées dans le domaine de la reconnaissance de formes ou de l'apprentissage mais avec une philosophie différente [Gascuel and Danchin, 1986]).

Pour Waterman, un modèle, qu'il appelle un motif consensuel ('consensus pattern'), est un mot défini sur le même alphabet que celui des chaînes à comparer. L'application la plus immédiate de sa méthode concerne les séquences d'acides nucléiques, mais elle est également valable pour les protéines. Par souci de clarté, nous la décrivons sur l'exemple des séquences nucléiques.

L'idée de l'algorithme est très simple. Elle consiste à engendrer tous les modèles possibles, c'est-à-dire tous les mots possibles de taille k sur $\Sigma = \{A, C, G, T \text{ ou } U\}$ pour un k donné, et à rechercher ces mots en autorisant un certain nombre d'erreurs (substitutions et éventuellement trous). Un mot u dans une des chaînes est ainsi une occurrence d'un modèle m si u présente au plus e erreurs avec m. Cette approche consiste donc à rechercher les modèles L-présents dans un ensemble de chaînes suivant la définition 3.5.2. Si aucune contrainte supplémentaire n'était imposée, les groupes de mots similaires trouvés correspondraient à ceux de la définition 3.5.3.

Comme Waterman doit engendrer tous les modèles possibles afin de localiser ceux qui l'intéressent (nous verrons bientôt quels sont ceux qu'il va considérer comme intéressants), il est limité dans la longueur k des modèles qu'il peut identifier. En pratique, cette longueur est en général fixée entre 4 et 6 pour les acides nucléiques. Pour accélérer les calculs et parce que rechercher des modèles (et leurs occurrences) relativement petits sur tout l'espace des chaînes peut produire beaucoup de 'bruit', Waterman limite de plus sa recherche à l'intérieur d'une fenêtre de taille W placée sur chacune des chaînes. Bien sûr, cela suppose que ces dernières on pu être pré-alignées sur la base d'un critère biologique quelconque (par exemple, le début de transcription de l'ADN en ARN [Galas *et al.*, 1985]). Le choix de la taille W est un compromis entre efficacité et sensibilité de la recherche. La méthode consiste à chercher pour chaque modèle possible de longueur k les mots de même longueur (si seules les substitutions sont autorisées) ou de longueur égale plus ou moins e (si les trous sont également permis) qui se situent à l'intérieur de la fenêtre et qui présentent au plus e erreurs avec le modèle. Chacun des modèles m reçoit alors un score donné par la formule suivante :

$$score(m) = \sum_{0 \leq d \leq e} \lambda_d q_{md}$$

où q_{md} est le nombre de chaînes pour lequel la meilleure occurrence de m se trouve à une distance d de m et λ_d pour $0 \leq d \leq e$ est un ensemble de poids fixés à l'avance. En général,

nous avons :

$$\lambda_d = 1 - \frac{d}{k}.$$

Le modèle ayant le meilleur score (en cas d'égalité, un modèle est choisi arbitrairement) constitue le modèle 'intéressant' pour cet emplacement de la fenêtre. Celle-ci est alors glissée vers la droite et la recherche d'un autre modèle se poursuit. Le glissement de la fenêtre peut être effectué de diverses manières. Par exemple, celle-ci peut se déplacer d'une position seulement vers la droite sur chacune des chaînes [Galas *et al.*, 1985] [Waterman, 1989]. Les modèles trouvés à chaque glissement peuvent alors se superposer ou même être identiques du point de leurs 'noms' et/ou de leurs ensembles d'occurrences [Galas *et al.*, 1985] [Waterman, 1989] (voir la figure 5.1) bien que l'on puisse également exiger que cela ne soit pas le cas [Waterman, 1986]. Dans une autre version [Waterman, 1986], le modèle trouvé par rapport au placement initial de la fenêtre constitue un premier point d'ancrage des chaînes, qui sont alors alignées sur ce modèle (ses occurrences), puis la fenêtre est glissée vers la droite pour venir se placer à la première position qui suit ces occurrences sur les chaînes. La recherche d'un nouveau modèle dont les occurrences sont forcément différentes de celles du premier recommence alors (voir la figure 5.2). Outre le fait qu'il permet de repérer certains consensus, ce second processus produit ainsi naturellement un alignement (heuristique) des chaînes.

L'extension de cette méthode à la comparaison de séquences de protéines est immédiate. Les modèles sont toujours des mots (en général de longueur 4) définis cette fois sur l'alphabet des acides aminés, et le score d'un modèle correspond à la somme des scores de tous les alignements du modèle avec sa meilleure occurrence située à l'intérieur de la fenêtre sur chaque chaîne [Waterman, 1989]. Il correspond donc à un score Étoile Centrale (voir la section 3.4.1.5 du chapitre 3).

La complexité en temps de l'algorithme est $O(n.N.W^2.B)$, où n et N sont toujours la longueur moyenne et le nombre des chaînes respectivement et où B est une fonction de $\mid \Sigma \mid^k$. Suivant l'implémentation qui en est faite, l'algorithme peut avoir une complexité en espace très réduite, en $O(k)$.

5.2.2 Des modèles comme des mots à positions fixes et positions variables : l'algorithme de Smith

L'idée d'un modèle n'apparaît que de manière implicite dans l'article de Smith destiné à la comparaison de protéines [Smith *et al.*, 1990]. Il définit ainsi un modèle, qu'il appelle un motif, comme étant composé d'un arrangement de deux (ou plus de deux) acides aminés séparés entre eux par des distances fixes. Par exemple, $m = A...Q.H..I$ est un modèle (motif) composé de 4 acides aminés (A, Q, H, I) séparés les uns des autres par une distance égale à 3, 1 et 2 acides aminés respectivement. Un mot dans une chaîne est une occurrence du modèle m s'il peut lui-même être écrit sous la forme $A...Q.H..I$ où les points représentent n'importe quel acide aminé.

L'algorithme de Smith consiste, étant donné deux paramètres k et d (des entiers non nuls), à engendrer tous les modèles de la forme $a_1 d_{12} a_2 d_{23} a_3 ... d_{(k-s)k} a_k$ où $a_i \in \Sigma$ et $0 \leq d_{(d-i)i} \leq d$, puis à rechercher l'occurrence de ces modèles dans les chaînes. Les a_i correspondent aux positions fixes des modèles et les acides aminés entre deux d'entre eux aux positions variables. Pour réaliser cette recherche, Smith utilise un tableau de taille $\mid \Sigma \mid^k d^{k-1}$ où tous les modèles possibles sont stockés. Il n'a besoin alors d'effectuer qu'un seul passage sur chacune des chaînes afin de repérer les occurrences des modèles, avec un arrêt de durée d^{k-1} sur chaque position. La complexité en temps de l'algorithme est ainsi $O(max\{n.N.d^{k-1}, \mid \Sigma \mid^k .d^{k-1}$(initialisation du tableau)$\})$ et celle en espace $O(\mid \Sigma \mid^k .d^{k-1} + n.N.d^{k-1})$, le dernier facteur indiquant que,

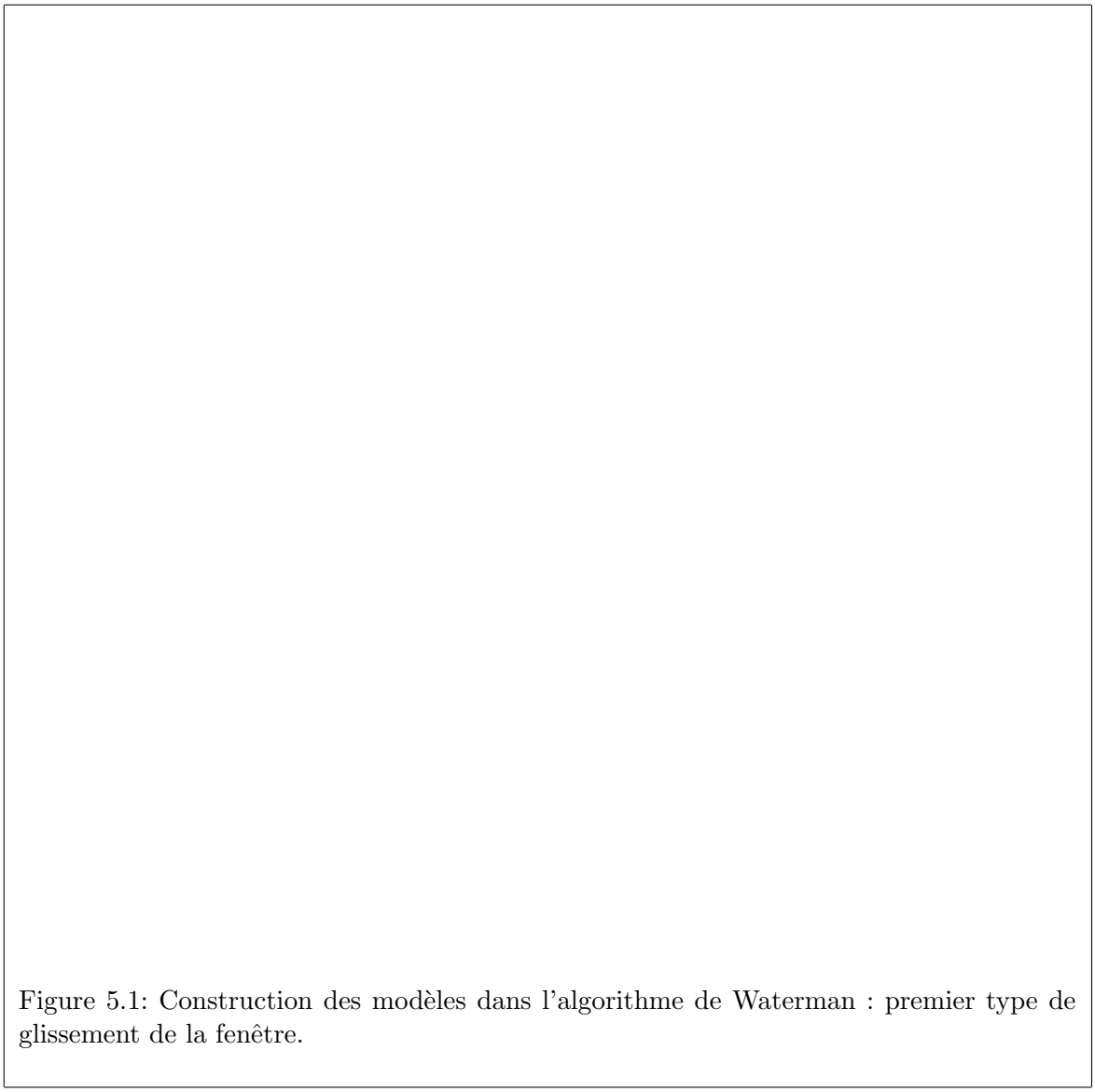


Figure 5.1: Construction des modèles dans l'algorithme de Waterman : premier type de glissement de la fenêtre.

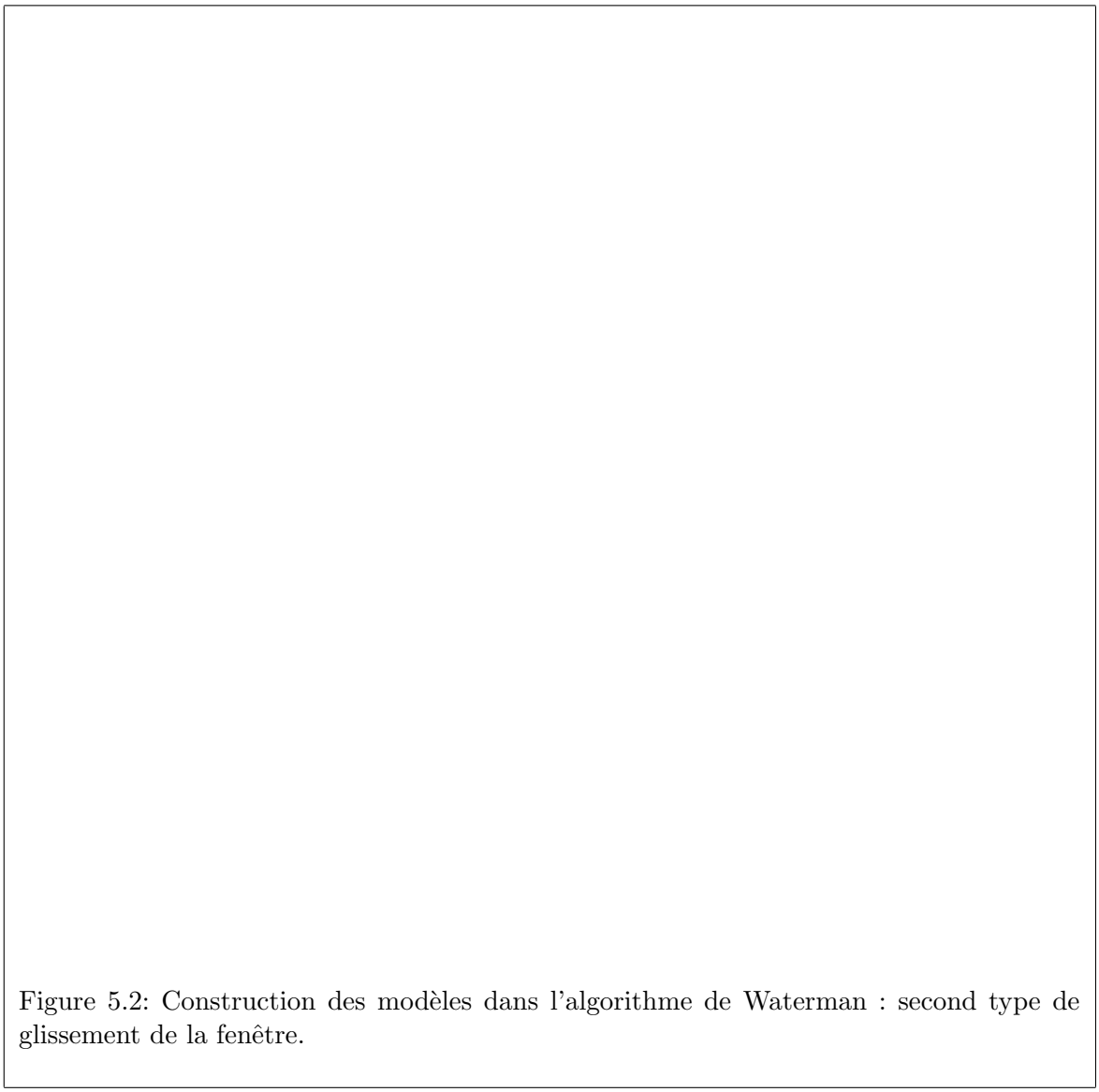


Figure 5.2: Construction des modèles dans l'algorithme de Waterman : second type de glissement de la fenêtre.

dans la pire des hypothèses (si les chaînes sont composées du même symbole), chaque position peut faire référence à d^{k-1} mots appartenant à des modèles différents. Il est clair qu'une telle approche n'est praticable que pour des petites valeurs de k (en général 3 ou 4) et de d (en général 10). De plus, afin de limiter le nombre de modèles trouvés, seuls ceux satisfaisant une contrainte de quorum sont retenus, et parmi ceux-là, seuls ceux ayant le meilleur score sont reportés par l'algorithme, où le score d'un modèle correspond à un score SP de ses occurrences. Par contre, la recherche des modèles se fait sur tout l'espace des chaînes simultanément et non plus uniquement à l'intérieur d'une fenêtre de taille fixe comme cela est le cas pour Waterman. Aucune de ces deux méthodes n'est cependant assez flexible, soit dans la définition de ressemblance entre mots, soit dans sa réalisation pratique (en particulier elles ne permettent de résoudre que le Problème 1), et le reste de ce chapitre est consacré à l'introduction de méthodes indirectes de comparaison plus générales et plus efficaces.

5.3 Aperçu général

5.3.1 La construction par récurrence des modèles

5.3.1.1 La construction comme un parcours dans un arbre

5.3.1.1.1 L'arbre de tous les modèles et celui des modèles valides

Nous allons examiner ce que résoudre le problème général signifie en considérant le cas le plus simple où les modèles sont des mots (ou des sous-ensembles unitaires de l'alphabet) et où aucune erreur (substitution ou trou) n'est autorisée entre un modèle et ses occurrences. Le problème est alors simplement de localiser tous les mots répétés exactement dans $N \geq 1$ chaînes, où le nom des modèles associés à chaque ensemble de mots est simplement le nom de l'un quelconque de ces mots. C'est donc le problème traité par KMR (section 4.2.3.4). Pour simplifier, nous supposons également que les $N \geq 1$ chaînes que nous voulons comparer sont concaténées en une seule (voir l'extension de KMR réalisée par Landraud, section 4.2.3.4) et nous oublions pour l'instant les détails d'implémentation permettant d'éliminer les mots répétés à cheval sur deux chaînes et de vérifier sur laquelle des chaînes est effectivement présent un modèle.

Dans ces conditions, considérons l'arbre de tous les modèles possibles de longueur au plus k (un exemple d'un tel arbre est donné dans la figure 5.3 pour $k = 3$ et un alphabet constitué de 2 symboles). Supposons que chaque nœud de l'arbre représente un récipient contenant les positions d'une chaîne s où le modèle qui donne son nom au nœud est présent de manière exacte. Si s était une chaîne aléatoire et infinie, chaque nœud comporterait au moins une telle position. Et trouver tous les modèles, ainsi que leurs occurrences, présents au moins une fois dans s correspondrait à donner la liste de toutes les feuilles de cet arbre (c'est-à-dire à donner leur noms et à énumérer le contenu de leurs récipients).

Bien sûr, cela est un cas extrême, puisque s est en pratique une chaîne de longueur finie. De plus, s représente en fait le plus souvent un ensemble de chaînes et les modèles qui nous intéressent en général sont ceux qui satisfont une contrainte de quorum q. Nous ne devons ainsi fournir que la liste des feuilles de l'arbre correspondant aux modèles qui sont effectivement présents au moins q fois dans la chaîne (nous appelons ces modèles des modèles valides). En prenant à nouveau comme exemple un alphabet à deux symboles, nous montrons dans la figure 5.4 à quoi ressemble cet arbre pour un ensemble de trois chaînes et pour $q = 3$. Observons qu'il s'agit d'un arbre déjà beaucoup plus élagué que celui de la figure 5.3. Il serait encore plus élagué pour des valeurs plus élevées de k.

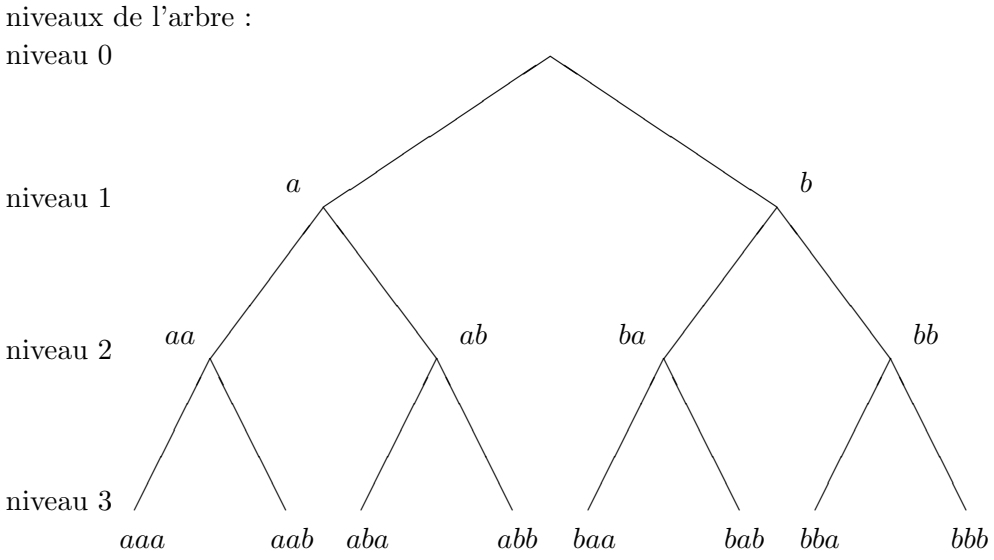


Figure 5.3: Arbre de tous les modèles de longueur au plus 3 sur un alphabet à 2 symboles.

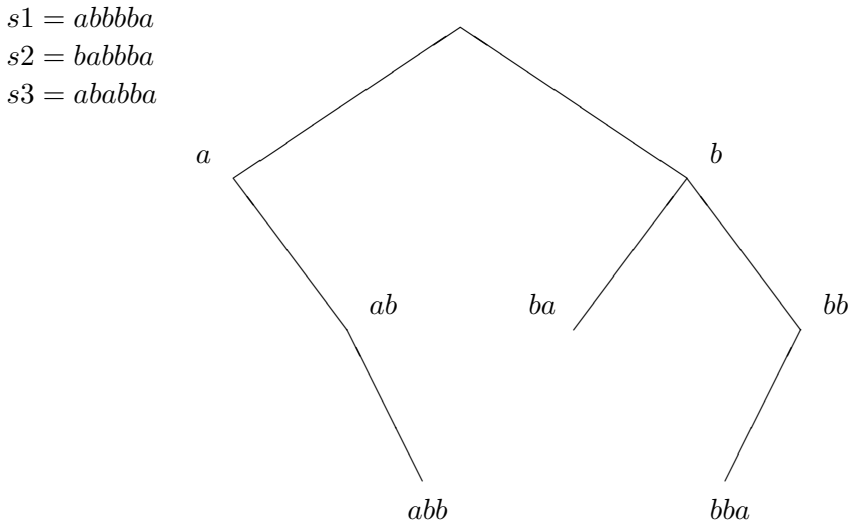


Figure 5.4: Arbre de tous les modèles valides de longueur au plus 3 sur un alphabet à 2 symboles pour un ensemble de 3 chaînes et avec $q = 3$.

Le degré d'élagage de l'arbre va en fait dépendre non seulement de k mais aussi des chaînes, de la valeur attribuée à q et de l'autorisation ou non de substitutions et d'erreurs. D'une façon générale cependant, l'arbre des modèles valides de longueur k va contenir beaucoup moins de nœuds que l'arbre de tous les modèles.

Ces considérations sont très importantes car il est évident que pour obtenir le résultat désiré, c'est-à-dire fournir une liste des feuilles de l'arbre de tous les modèles valides (par opposition à tous les modèles possibles), nous devons traverser cet arbre d'une manière ou d'une autre. En effet, la seule façon que nous aurions de connaître directement le contenu d'un récipient à un nœud quelconque (excepté ceux localisés au premier niveau) serait de construire une table indexée ou de hachage comme cela est fait par Leung, Martinez ou Santibanez (voir la section 4.2.3.2), ou encore Waterman ou Smith (section 5.2). Cela pourrait être très inefficace (puisque tous les nœuds ne sont pas présents) et très coûteux en espace pour des arbres avec plus de 3 ou 4 niveaux. Une façon plus naturelle et performante de calculer le contenu d'un récipient à n'importe quel niveau en-dessous du premier est donc d'utiliser comme information le contenu des récipients des nœuds situés à des niveaux moins élevés dans l'arbre (ceux qui se trouvent plus près de la racine de l'arbre) et qui auraient été déterminés dans une étape antérieure. Dans le cas d'une relation d'identité et lorsqu'aucune erreur n'est autorisée, un arbre ou un automate des suffixes pourraient également être employés mais ces structures sont relativement lourdes à établir alors que l'arbre des modèles est parcouru sans jamais être effectivement construit. Une partie de l'information que cet arbre représente est bien sûr maintenue en mémoire par les algorithmes comme KMR/KMRC ou ceux que nous allons proposer ici, nous verrons un peu plus loin quelle partie exactement, mais celle-ci ne compose qu'un sous-ensemble réduit de toute l'information qui doit être stockée dans une structure comme un arbre ou un automate des suffixes. Par ailleurs, la construction et l'utilisation de ces dernières structures ne permet pas facilement, du moins pour l'instant, de tenir compte de relations de similarité plus complexes (substitutions conservatives modélisées par une couverture, erreurs, couverture combinatoire pondérée, similarité basée sur des mots).

Observons que l'arbre des modèles valides n'est pas connu a priori. Cependant, il peut être facilement déterminé en réalisant un parcours de l'arbre de tous les modèles possibles qui s'arrête aussitôt qu'un nœud est rencontré dont le modèle correspondant ne vérifie pas la contrainte de quorum. En effet, si le modèle qui donne son nom à un nœud x ne vérifie pas ce quorum, aucun modèle de plus grande longueur l'ayant comme préfixe ne peut lui-même vérifier le quorum. Le sous-arbre de l'arbre de tous les modèles possibles ayant le nœud x comme racine peut donc être coupé. De cette manière, ce qui est réellement parcouru est l'arbre des modèles valides seulement, plus éventuellement $\mid \Sigma \mid$ nœuds (dans ce cas les modèles sont des mots) à chaque feuille de l'arbre comme des 'regards-en-avant' (en anglais, des 'look-aheads'). En fait, nous verrons qu'il est possible dans certains cas de lancer moins de regards en avant.

Les modèles et leurs ensembles d'occurrences sont ainsi trouvés récursivement.

5.3.1.1.2 Le parcours dans l'arbre : en largeur ou en profondeur

Considérons toujours le cas le plus simple où les modèles sont des mots et la relation de ressemblance entre ceux-ci et leurs occurrences est l'identité (aucune erreur n'est donc autorisée). Nous venons de voir que parcourir l'arbre des modèles valides (ceux qui vérifient un certain quorum q) afin de fournir la liste des feuilles de niveau k ou k_{max} (donner leurs noms et énumérer le contenu de leurs récipients) correspond exactement à ce que fait KMR et les classes d'équivalence que l'algorithme établit sont implicitement associées à des modèles.

La façon la plus efficace de traverser cet arbre pour atteindre ses feuilles utilise alors, dans

ce cas, le lemme 4.2.1 ou le lemme 4.2.2 de KMR avec $a = b$ tant que cela est possible, puis le lemme 4.2.1 avec $a \neq b$. Utiliser le lemme 4.2.1 ou le lemme 4.2.2 avec $a = b$ correspond en fait à réaliser une exploration en largeur de l'arbre où seuls les niveaux qui sont des puissances de 2 sont visités (c'est-à-dire le contenu de leurs récipients calculés).

Lorsque les modèles sont maintenant explicitement introduits ainsi que des définitions plus flexibles de la ressemblance entre eux et les mots dans les chaînes, des lemmes souvent très semblables à ceux de KMR (ou KMRC) peuvent être établis permettant de calculer l'ensemble des occurrences d'un modèle quelconque à partir des ensembles d'occurrences de modèles strictement plus courts. Ces lemmes seront donnés plus loin mais il est relativement aisé de voir dès à présent que, lorsque des erreurs sont autorisées (définitions 3.5.3 et 3.5.8 de ressemblance) ou une pondération introduite sur les éléments d'une couverture (définition 3.5.12), seule une forme modifiée du lemme 4.2.1 peut être employée pour construire les modèles de longueur k à partir de ceux de longueur $k' < k$. Nous ne pouvons plus en effet réaliser une superposition de deux occurrences afin d'en obtenir une troisième qui garantisse que cette dernière continuera de vérifier les mêmes propriétés par rapport à son modèle que les deux premières par rapport aux leurs (nombre d'erreurs ou pondération des éléments de la couverture en-dessous de la valeur maximale permise — voir la figure 5.5). En revanche, lorsqu'un système de score des sous-mots d'une occurrence est utilisé (définition 3.5.15), c'est une forme modifiée du lemme 4.2.2 qui est seule applicable, à condition que b soit strictement plus petit que a (voir la figure 5.6).

Dans le premier cas, nous pouvons donc toujours doubler la longueur des modèles construits à chaque étape, et dans le second cas nous pourrions presque la doubler (i.e. nous pourrions l'augmenter de $k - w + 1$ où w est la longueur des sous-mots dont le score doit être au-dessus d'un certain seuil). Cependant lorsqu'une telle opération de dédoublement n'est plus possible et qu'il faille utiliser la superposition (lemme 4.2.2) pour continuer à étendre les modèles, alors la longueur de ces derniers ne peut plus être incrémentée que d'une unité à chaque fois jusqu'à ce que la longueur désirée soit atteinte, ou jusqu'à la plus grande longueur pour laquelle cette contrainte est encore vérifiée, sauf à garder en mémoire les ensembles d'occurrences de tous les modèles pour plusieurs longueurs de ces derniers.

De plus, si nous explorions l'arbre en largeur en avançant par puissance de 2 (ou presque), nous aurions à nouveau besoin d'obtenir et puis de stocker tous les ensembles d'occurrences de tous les modèles de longueur k qui se trouvent L/EL/ECL/w-présents au moins q fois dans les chaînes avant de pouvoir construire les modèles de longueur $2k$ (ou $2k - w + 1$). Lorsque la relation de ressemblance entre modèles et mots est très flexible (ce que nous souhaitons), les modèles de longueur petite (jusqu'à 4 ou 5) sont presque tous présents et chacun possède beaucoup d'occurrences. Un tel mode de parcours pourrait en conséquence exiger trop d'espace puisque tous les nœuds du niveau k sont nécessaires afin de construire les nœuds du niveau $2k$ (ou $2k - w + 1$).

Un moyen plus économique en mémoire pour construire ces modèles consiste donc à réaliser un parcours en profondeur de l'arbre des modèles. En effet, si au lieu de doubler la longueur des modèles à chaque étape, nous étendons chaque modèle séparément vers la droite d'une unité à la fois, alors pour obtenir les ensembles d'occurrences d'un modèle donné, nous n'avons besoin que de l'ensemble d'occurrences de son préfixe, plus les ensembles d'occurrences de tous les modèles de longueur 1 ou w. Cela correspond à un parcours dans l'arbre dont la complexité en temps présente un rapport $\frac{k}{\log k}$ avec le parcours réalisé 'à la façon KMR'. Comme k est en pratique petit (autour de 10-20), ce rapport est faible. Il est également relativement négligeable par rapport à $n.N$ (où n est la longueur moyenne des chaînes).

Quel que soit le mode de parcours adopté, celui-ci est bien sûr associé à une certaine opération qui doit être effectuée à chaque nœud x afin de pouvoir descendre vers ses nœuds-fils

Figure 5.5: Problème si le second lemme de KMR est utilisé (avec $b < a$) lorsque des erreurs ou une pondération des ensembles de la couverture sont introduites.

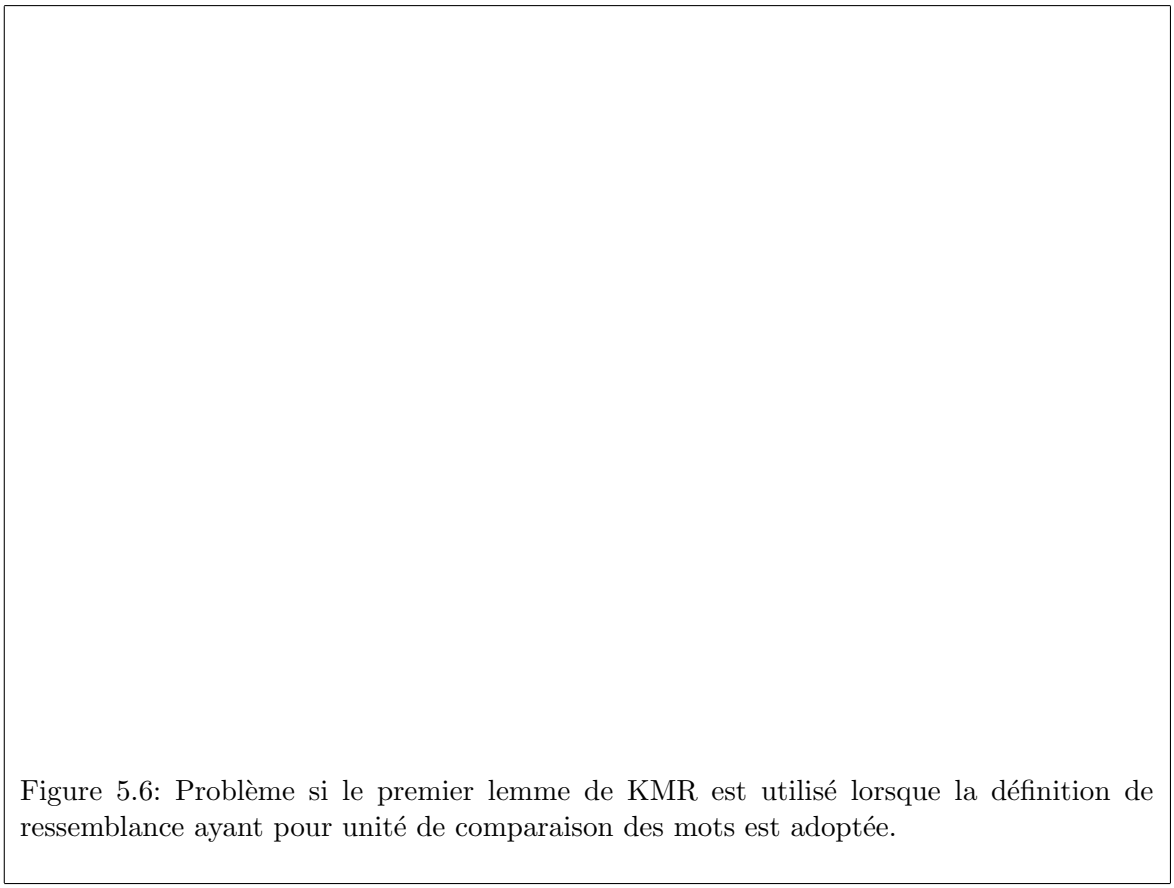


Figure 5.6: Problème si le premier lemme de KMR est utilisé lorsque la définition de ressemblance ayant pour unité de comparaison des mots est adoptée.

dans l'arbre des modèles. Cette opération peut être vue comme une attribution d'étiquettes, c'est ce que nous avons fait dans la section 4.2.3.4. Elle peut également être décrite comme un partitionnement d'ensembles (de positions dans les chaînes) lorsque la relation entre modèles et mots est l'identité ou une relation d'équivalence, ou comme un raffinement de ces mêmes ensembles dans tous les autres cas.

5.3.1.2 Le construction comme un partitionnement/raffinement d'ensembles

Considérons un parcours dans l'arbre des modèles valides. À nouveau ici, nous commençons par supposer que nous travaillons avec une relation de similarité entre modèles et mots qui est l'identité et que les chaînes sont concaténées en une seule, notée s. Supposons par ailleurs également que nous avons atteint un certain nœud x situé au niveau $k-1$. Cela signifie que nous venons de construire un modèle valide m (les modèles sont dans ce cas des mots) de longueur $k-1$, c'est-à-dire, nous venons de calculer son ensemble d'occurrences dans les chaînes. Construire le modèle de longueur k ayant m comme préfixe (i.e. établir quelles sont ses occurrences), et donc passer au niveau supérieur dans l'arbre, peut être fait en plaçant dans le récipient associé au nœud étiqueté par ma pour tout $a \in \Sigma$ les positions i du récipient associé à m pour lesquelles $s_{i+m} = a$. Chaque position d'une occurrence de m dans s va donc aller se placer dans un et un seul des récipients rattachés aux modèles ma de longueur k. Ce placement est simplement déterminé par le symbole a qui suit l'image u associée à chaque occurrence de m (voir la figure 5.7). Ce processus représente bien une opération de partitionnement, suivant Σ, réalisée sur l'ensemble des occurrences de m. Et traverser l'arbre des modèles correspond donc ainsi à effectuer une telle opération sur chaque nœud visité de l'arbre.

Lorsque la relation entre modèles et mots n'est plus l'identité, ni une relation d'équivalence, une position donnée d'un récipient situé à un certain niveau $k-1$ peut glisser vers plusieurs récipients différents du niveau k, ou même glisser plusieurs fois vers un même récipient de ce niveau. Des exemples de ces deux situations sont présentés dans les figures 5.8 et 5.9. Ce phénomène sera plus précisément décrit lorsque nous entrerons dans le détail de chacun des algorithmes. Nous y montrerons alors que dans certaines situations, ce problème de la 'multiplication' d'une position peut être en partie contourné. Parcourir l'arbre des modèles équivaut encore une fois à réaliser une certaine opération sur les contenus des récipients rattachés à chaque nœud de l'arbre, simplement l'opération n'est plus une partition mais fait référence à un raffinement des ensembles de positions associés aux modèles.

Observons que dans ce parcours dans l'arbre des modèles, il n'est pas toujours nécessaire de suivre un nombre de branches égal à $\mid \Sigma \mid$ (si les modèles sont des mots) ou p (si les modèles sont des produits cartésiens d'ensembles d'une couverture) à partir d'un nœud x vers ses nœuds fils. Cela a été suggéré déjà dans le cas d'une relation d'identité (il suffit de regarder quels symboles sont effectivement présents dans les chaînes), mais ce n'est pas la seule situation où une telle économie est possible. Cette remarque a un effet sur la complexité de l'algorithme et sera précisée plus loin.

Par contre, il paraît assez intuitif de considérer que le parcours dans l'arbre va nécessiter dans le pire des cas n étapes si les modèles ne sont rallongés que d'une unité à la fois, et qu'elle ne peut être réalisée en $\log n$ étapes que lorsqu'est adoptée l'idée de KMR de doubler la longueur des mots répétés/modèles à chaque fois (ce qui correspond à visiter uniquement les niveaux de l'arbre qui sont une puissance de 2).

Cette intuition est fausse. Dans le cas d'un partitionnement au moins (i.e., lorsque la relation est l'identité), il a été montré par Aho [Aho *et al.*, 1974] qu'une suite de n partionnements demande au plus $O(n.\log n)$ opérations. Aho observe en effet que le partitionnement d'un

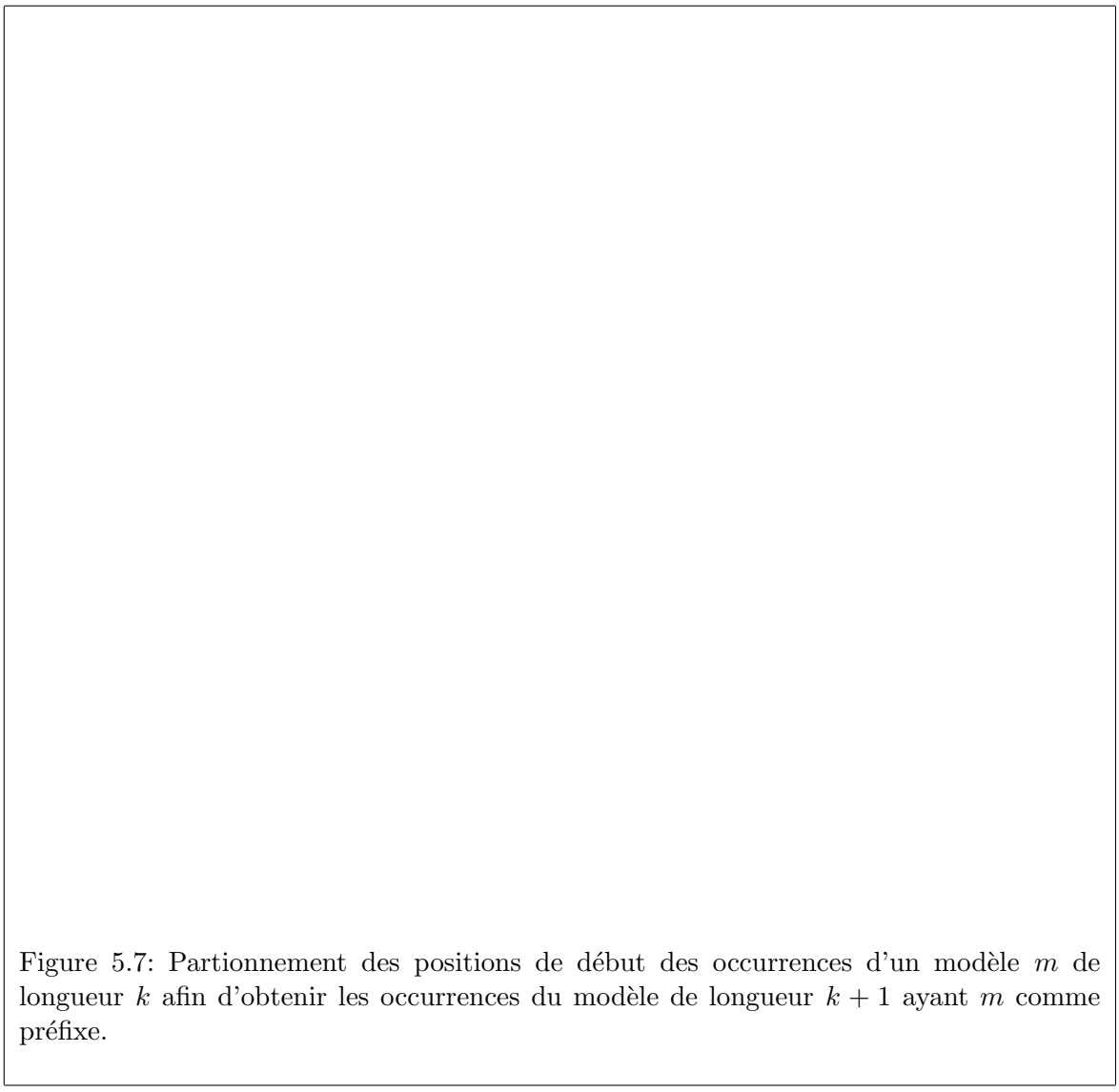


Figure 5.7: Partionnement des positions de début des occurrences d'un modèle m de longueur k afin d'obtenir les occurrences du modèle de longueur $k+1$ ayant m comme préfixe.

Figure 5.8: 'Multiplication' des positions : glissement vers plusieurs des récipients du niveau suivant de l'arbre.

Figure 5.9: 'Multiplication' des positions : glissement en plusieurs 'exemplaires' vers un (même) récipient du niveau suivant de l'arbre.

/* Entrée */
q : contrainte de quorum
k : une longueur fixe ou la plus grande possible
/* Sortie */
Tous les modèles de longueur k qui ont des occurrences dans au moins q des chaînes

Algorithmes pour résoudre le problème général :

1. Déterminer les ensembles d'occurrences de tous les modèles de longueur 1 (distance basée sur les symboles) ou de longueur w (distance basée sur des mots) qui vérifient la contrainte de quorum q;

2. Appliquer des opérations de partionnement ou de raffinement successives afin de déterminer les ensembles d'occurrences des modèles de longueurs croissantes qui vérifient la contrainte de quorum q jusqu'à ce que cette longueur soit égale à k (Problème 1) ou soit la plus longue possible (Problème 2).

En termes de parcours d'un arbre, cela peut être reformulé de la façon suivante:

Algorithmes pour résoudre le problème général :

1. Déterminer les ensembles d'occurrences de tous les modèles de longueur 1 (distance basée sur les symboles) ou de longueur w (distance basée sur des mots) qui vérifient la contrainte de quorum q;

2. Appliquer des opérations de partionnement ou de raffinement successives afin de traverser l'arbre des modèles et énumérer le contenu des récipients au niveau k (Problème 1) ou ceux au niveau le plus élevé (Problème 2).

Figure 5.10: Algorithme prototype pour résoudre le problème général.

ensemble en deux sous-ensembles peut en fait être réalisé en un temps qui est proportionnel au nombre d'éléments du plus petit des deux sous-ensembles uniquement. Crochemore et Cardon [Cardon and Crochemore, 1982] [Crochemore, 1981] ont prouvé alors que cette idée demeure valable même lorsqu'un ensemble est partitionné en $c > 2$ sous-ensembles : dans ce cas, le partitionnement demande un temps proportionnel au nombre d'éléments des $c - 1$ plus petits sous-ensembles.

Nous disposons maintenant des idées de base pour pouvoir indiquer un prototype de tous les algorithmes que nous présenterons dans la suite de ce chapitre.

5.3.2 Un premier prototype d'algorithme

L'algorithme prototype pour résoudre le problème général (voir section 5.1) et qui forme le squelette de tous les algorithmes que nous avons établis et introduisons dans ce chapitre est donné dans la figure 5.10.

Un premier aperçu intuitif de sa complexité est présenté dans la section suivante.

5.3.3 Complexité : un premier aperçu

Nous allons d'abord considérer la complexité en temps de la méthode que nous proposons. La forme récursive de l'algorithme prototype, qui sera retrouvée dans tous les algorithmes particuliers montrés plus loin, conduit à une formulation de sa complexité en temps qui est également récursive.

Ainsi la question principale que nous devons nous poser afin d'établir cette complexité est : supposons que nous ayons construit tous les modèles d'une certaine longueur $k-1$ pour $k \geq 1$, combien d'opérations devons-nous alors réaliser afin d'obtenir les modèles de longueur k à partir de ceux que nous connaissons?

Avec tout ce que nous venons de voir, il apparaît assez clairement que trois facteurs vont être déterminants dans la complexité en temps de l'algorithme. Le premier est bien sûr le nombre total de modèles de longueur $k-1$ et de leurs occurrences — notons ce nombre $N_{ModOcc}(k-1)$. Celui-ci est en général lié au produit du nombre total de modèles par le nombre maximal d'occurrences de chaque modèle, ce dernier nombre étant lui-même en rapport avec la longueur totale $n.N$ des chaînes. Le second facteur déterminant de la complexité est le nombre maximal de branchements à suivre à chaque étape dans le parcours de l'arbre — notons-le $N_{MaxBranches}$ — qui correspond au nombre de regards-en-avant qu'il faut lancer à chaque nœud, ou de branches par lesquelles il faut glisser pour passer d'un niveau au suivant. Ce facteur est en théorie indépendant de k. Enfin le troisième facteur est fonction de la longueur des modèles à rechercher — notons-le $f(L_M)$. Si nous appelons $CompTemps$ cette complexité, nous aurons alors dans tous les cas quelque chose de la forme :

$$CompTemps = N_{ModOcc}(k-1) \times N_{MaxBranches} \times f(L_M).$$

Observons que la contrainte de quorum q n'est pas prise en compte dans cette formule, du moins de façon directe. Elle pourrait l'être dans le calcul de $N_{ModOcc}(k-1)$, mais en fait son influence sur le nombre des modèles va dépendre très fortement des données et n'est donc pas considérée dans la valeur théorique (pire des cas) de la complexité.

Toute la difficulté maintenant d'établir une 'bonne' borne supérieure pour $CompTemps$ va dépendre de la façon de calculer ces divers facteurs, en particulier $N_{MaxBranches}$ et $N_{ModOcc}(k-1)$, ainsi que le rapport de ce dernier avec le nombre de modèles possibles et la longueur des chaînes.

Une première manière de procéder afin de calculer une borne supérieure à $N_{ModOcc}(k-1)$ est de calculer tout simplement le nombre total de modèles pouvant être présents dans les chaînes. Cette approche est naïve car la seule chose que nous pouvons dire de cette borne dans l'absolu est qu'elle est égale à $\mid \Sigma \mid^{k-1}$ ou p^{k-1} suivant que les modèles de longueur $k-1$ sont définis comme étant des mots ou des produits d'ensembles d'une couverture comportant p éléments.

Une meilleure façon de faire consiste à prendre le problème à l'envers, c'est-à-dire à considérer les positions dans les chaînes et à calculer le nombre maximal de modèles auxquelles chacune de ces positions (indicatives du début d'une occurrence dans les chaînes) peut appartenir — notons ce nombre $N_{Modi}(k-1)$. Il dépend de la définition de ressemblance entre modèles et mots adoptée, mais est en général bien plus petit que $\mid \Sigma \mid^{k-1}$ ou p^{k-1} (dans le cas de KMR en particulier, $N_{Modi}(k-1)$ est égal à 1). Comme il y a au plus $n.N$ positions dans les $N \geq 1$ chaînes de l'ensemble à comparer, nous devrions alors obtenir :

$$N_{ModOcc}(k-1) \leq n.N \times N_{Modi}(k-1).$$

Cela n'est cependant pas encore tout à fait exact. Souvenons-nous (voir la section 5.3.1.2 ainsi que les figures 5.8 et 5.9) qu'une position peut se trouver présente plusieurs fois dans un même

récipient de l'arbre des modèles (pour des occurrences/images différentes). La vraie borne supérieure pour $N_{ModOcc}(k-1)$ est donc :

$$N_{ModOcc}(k-1) \leq n.N \times F_{MultiPos} \times N_{Modi}(k-1)$$

où $F_{MultiPos}$ est un facteur multiplicatif qui indique le nombre de fois où une position peut appartenir à un même modèle (pour des images différentes) et dépend de la définition de ressemblance choisie. Nous verrons par la suite que ce facteur est en général petit par rapport à $N_{Modi}(k-1)$. Il n'est en fait différent de 1 que lorsque les trous sont permis dans une définition de ressemblance.

Considérons maintenant le facteur $f(L_M)$. Celui-ci est égal soit à la longueur des modèles recherchés, soit au log de cette longueur suivant que l'arbre est parcouru en longueur ou en largeur (et lorsque de plus seuls les nœuds situés aux niveaux qui sont des puissances de 2 sont visités).

Reste alors à calculer une borne supérieure pour le facteur $N_{MaxBranches}$. Pour cela, imaginons que nous avons atteint un certain nœud x situé au niveau $k-1$ dans l'arbre des modèles. Nous souhaitons voir si le modèle associé au nœud x peut être prolongé en un modèle de longueur k. Si la relation de similarité entre modèles et occurrences (mots dans les chaînes) est l'identité, alors une possibilité serait de descendre le long de $\mid \Sigma \mid$ branches à partir du nœud x pour vérifier si ma pour $a \in \Sigma$ demeure un modèle valide. Comme nous l'avons fait observé, il est possible dans ce cas particulier d'être plus efficace que cela puisqu'en pratique, nous n'avons besoin de développer que les branches étiquetées a issues du nœud x pour lesquelles a apparaît au moins une fois dans les chaînes à la suite d'une des occurrences de m. Ce fait est illustré dans la figure 5.11. Il demeure valable lorsque la ressemblance entre modèles et mots dans les chaînes est une relation d'équivalence, une relation unique non transitive comme dans Soldano [Soldano *et al.*, 1995] (voir sections 3.4.3.2 et 4.2.3.4), ou encore est basée sur une couverture (section 3.5.2.1.2) si les erreurs ne sont pas autorisées, ou sur une couverture combinatoire pondérée (section 3.5.2.1.3). Cela signifie qu'une position i appartenant à $N_{Modi}(k-1)$ modèles de longueur $k-1$ ne peut 'donner naissance' (i.e. ne peut aller appartenir) qu'à 1 ou g modèles de longueur k respectivement (où g mesure le degré de non-transitivité d'une couverture — voir 3.2.15). Ainsi le nombre total de branches liant le niveau $k-1$ au niveau k, et donc le nombre total d'opérations à réaliser pour construire les modèles de longueur k à partir de ceux de longueur $k-1$, est au plus égal à $1 \times N_{Modi}(k-1)$ ou $g \times N_{Modi}(k-1)$.

Lorsque des erreurs sont permises dans la comparaison d'un modèle avec les mots dans les chaînes, que ces erreurs soient comptées (sections 3.5.2.1.1 et 3.5.2.1.2) ou pondérées (section 3.5.2.2), une telle économie n'est plus réalisable. Examinons d'abord pourquoi lorsque les erreurs sont comptées (la relation entre modèles et mots mesure une distance). Il est assez aisé de voir qu'il suffit alors qu'il y ait une seule occurrence d'un modèle qui n'ait pas encore atteint son 'quota' d'erreurs permises (et cela sera le plus souvent le cas) pour que son prolongement d'une unité vers la droite puisse être une occurrence du modèle également rallongé d'une unité, quel que soit le symbole (de l'alphabet ou de la couverture) par lequel ce modèle est étendu. Cette constatation est illustrée dans la figure 5.12. Dans ce cas, toutes les $\mid \Sigma \mid$ ou p branches possibles doivent être développées pour cette occurrence-là. Éventuellement, une (ou plusieurs) de ces branches conduira à un modèle qui n'est plus valide et pourra être alors coupée. Mais dans le pire des cas, $\mid \Sigma \mid \times N_{Modi}(k-1)$ ou $p \times N_{Modi}(k-1)$ branches doivent être examinées entre les niveaux $k-1$ et k.

Lorsque la relation de ressemblance entre modèles et mots est celle de la définition 3.5.15 ayant pour unité de comparaison des mots, cette observation demeure valable dans la mesure où il suffit qu'une occurrence d'un modèle possède une bonne 'réserve' en termes du score de

Figure 5.11: Branches à explorer à partir d'un nœud de l'arbre des modèles lorsque la notion de similarité ne permet pas d'erreurs (sauf les substitutions conservatives).

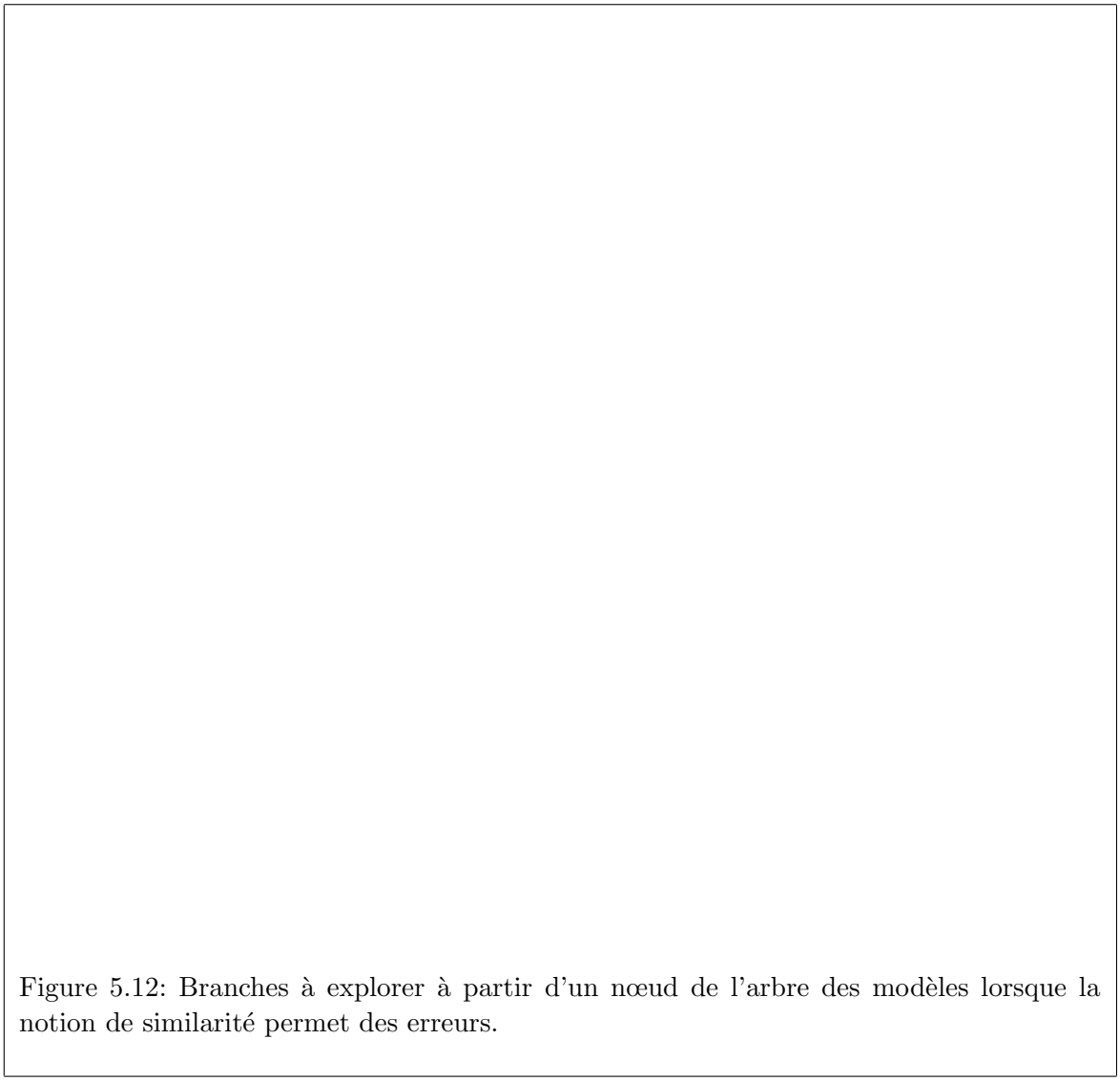


Figure 5.12: Branches à explorer à partir d'un nœud de l'arbre des modèles lorsque la notion de similarité permet des erreurs.

son dernier sous-mot avec le sous-mot correspondant du modèle de longueur $k-1$ pour qu'elle puisse 'donner naissance' à $\mid \Sigma \mid$ autres modèles de longueur k.

Ainsi, nous devrions avoir :

- $N_{MaxBranches} = 1$ ou g si les définitions de ressemblance adoptées sont les définitions 3.5.3 ou 3.5.8 avec $e = 0$, ou la définition 3.5.12;
- $N_{MaxBranches} = \mid \Sigma \mid$ ou p autrement.

À nouveau cette formule n'est pas tout à fait exacte, elle ne compte pas tout. Un facteur multiplicatif va intervenir encore une fois, que nous notons $F_{MultiBranche}$, et dont la raison est toujours liée aux trous. Considérons en effet le cas d'une définition de ressemblance basée sur une distance de Levenshtein. Il est assez facile de voir que, lorsque les trous sont autorisés, chaque occurrence d'un modèle de longueur $k-1$ va pouvoir engendrer potentiellement plusieurs occurrences du modèle de longueur k obtenu à partir du précédent (si le nombre

d'erreurs de l'occurrence initiale par rapport au modèle était encore nul). Certaines de ces occurrences sont en fait 'fausses' dans la mesure où le nombre d'erreurs qui va se trouver associé à elles par cette opération ne représente pas leur vrai distance au modèle étendu mais cela ne peut être déterminé qu'a posteriori et l'opération doit donc être comptabilisée malgré tout. Ce phénomène de multiplication d'une occurrence lors de la descente d'un niveau dans l'arbre sera plus précisément décrite dans la section 5.4.2.1. Il a été succintement illustré sur un exemple dans la figure 5.9. Il permet de voir qu'en réalité $F_{MultiBranche} \times \mid \Sigma \mid \times N_{Modi}(k-1)$ ou $F_{MultiBranche} \times p \times N_{Modi}(k-1)$ opérations sont nécessaires pour passer du niveau $k-1$ au niveau k de l'arbre des modèles, où $F_{MultiBranche}$ indique le nombre d'occurrences de longueur k qu'une occurrence de longueur $k-1$ peut engendrer. Comme nous l'avions mentionné (section 5.3.1.2), une autre façon de voir les choses est de considérer ce facteur comme représentant le nombre de fois où une occurrence peut 'glisser' le long d'une même branche de l'arbre des modèles.

Pour les mêmes raisons, un facteur multiplicatif semblable à celui que nous venons de décrire doit être appliqué lorsque la définition 3.5.15 de ressemblance est adoptée si les trous sont permis.

Ainsi, la valeur de $N_{MaxBranches}$ corrigée par ce facteur indique en fait dans tous les cas le produit du nombre maximal de branches par le nombre maximal de 'glissades' qu'il faut effectuer par branche et nous avons :

- $N_{MaxBranches} \leq 1$ ou g si les définitions de ressemblance adoptées sont les définitions 3.5.3 ou 3.5.8 avec $e = 0$, ou la définition 3.5.12;
- $N_{MaxBranches} \leq F_{MultiBranche} \times (\mid \Sigma \mid \text{ou } p)$ autrement.

La complexité en temps de l'algorithme prototype (et des algorithmes particuliers) peut donc être écrite sous la forme :

$$CompTemps \leq \underbrace{n.N \times F_{MultiPos} \times N_{Modi}(k-1)}_{\mathrm{N_{ModOcc}(k-1)}} \times \underbrace{\left\{ \begin{array}{l} 1 \textit{ ou } g \\ F_{MultiBranche} \times (\mid \Sigma \mid \textit{ ou } p) \end{array} \right.}_{\mathrm{N_{MaxBranches}}} \times \underbrace{\left\{ \begin{array}{l} k \\ \log k \end{array} \right.}_{\mathrm{f(L_M)}}.$$

Considérons maintenant la complexité en espace de l'algorithme. Celle-ci va également être composée d'un certain nombre de facteurs déterminants. Ceux-ci sont cependant différents suivant que l'arbre est parcouru en largeur ou en profondeur.

Dans le cas d'un parcours en largeur, nous pouvons profiter de tous les calculs que nous venons de faire afin de déterminer la complexité en temps de l'algorithme. En effet, ce qu'il faut compter ici est le nombre total de modèles et de leurs occurrences à un niveau k de l'arbre puisque c'est l'information qu'il nous faut stocker pour construire les modèles de longueur $k+1$ ou $2k$. Ce nombre correspond à $N_{ModOcc}(k)$. Si nous notons $CompEspace_{Largeur}$ la complexité en espace de l'algorithme par rapport à un parcours en largeur de l'arbre des modèles, nous avons alors :

$$CompEspace_{Largeur} \leq n.N \times F_{MultiPos} \times N_{Modi}(k).$$

Lorsque l'arbre est parcouru en profondeur, nous n'avons besoin de connaître que les modèles de longueur 1 plus tous ceux qui sont préfixes du modèle que nous sommes en train de construire. En fait, il est possible de ne conserver que ces derniers, les modèles de longueur 1 pouvant être facilement retrouvés (reconstruits) à chaque fois (en un temps $O(1)$). Si nous notons $CompEspace_{Longueur}$ la complexité en espace de l'algorithme par rapport à un parcours en profondeur, nous avons donc :

$$CompEspace_{Longueur} \leq n.N \times F_{MultiPos} \times (k-1)$$

puisqu'un modèle de longueur k a $k-1$ préfixes.

Dans ce qui suit, nous n'aurons plus qu'à déterminer les bornes supérieures pour les valeurs de $N_{Modi}(k-1)$, $F_{MultiPos}$ et $F_{MultiBranche}$ (qui vont dépendre de chaque l'algorithme).

5.4 Description détaillée

5.4.1 Introduction générale et conventions de notation

Nous allons présenter cette fois une visite détaillée de chacun des algorithmes composant notre approche.

Tous les algorithmes s'appuient sur un ou deux lemmes et sur une propriété permettant de construire les modèles (et donc de parcourir l'arbre) par longueurs croissantes. Ils sont en général donnés pour un parcours quelconque (en largeur ou en profondeur), mais ont été implémentés dans tous les cas pour un parcours en profondeur.

Dans tout ce qui suit, nous considèrerons que l'ensemble de $N \geq 1$ chaînes $s1, s2, ..., sN$ que nous souhaitons comparer est concaténé en une seule chaîne notée $s = s1s2...sN$. En fait, un symbole spécial sépare chacune des chaînes dans s, ceci afin de permettre d'éliminer facilement durant le déroulement de l'algorithme les mots qui se trouvent à cheval sur deux chaînes. De plus, un tableau auxiliaire de taille N conserve les positions de début (ou de fin) de chacune des chaînes $s1, s2, ..., sN$ dans la chaîne s. Comme les positions des mots dans s sont maintenues ordonnées dans les ensembles d'occurrences des modèles (cet ordre découle naturellement de l'implémentation), il est aisé grâce à ce tableau de s'assurer après sa construction qu'un modèle vérifie bien la contrainte de quorum.

Il est important de noter que chaque algorithme, ainsi que chaque version à l'intérieur d'un même ensemble d'algorithmes, est accompagnée d'une analyse de sa performance en pratique (dans l'absolu ou relative à celle des autres versions). En effet, le temps d'exécution observé en pratique va dépendre fortement de la valeur attribuée à la contrainte de quorum et, à travers cette valeur, des données. Lorsqu'il s'agira de résoudre le Problème 2 (trouver les modèles les plus longs), ces données vont également avoir en retour une influence sur la valeur de k_{max} et donc sur la complexité, théorique et pratique. C'est la raison pour laquelle donner seulement la formule pour cette complexité ne suffit pas à rendre compte du comportement effectif de l'algorithme. Diverses courbes sont donc à chaque fois présentées qui permettent de combler cette lacune. Pour les motifs mentionnés, il était important de fournir ces courbes par rapport à de 'vrais exemples' biologiques, mais notre propos n'est pas pour l'instant de discuter la signification des résultats obtenus en tant que tels. Cela ne sera fait en détail que dans le chapitre 6.

5.4.2 Les algorithmes utilisant des symboles comme unités de comparaison

5.4.2.1 Distance de Levenshtein classique ou ensembliste

5.4.2.1.1 Introduction

Le premier algorithme que nous allons présenter utilise des symboles comme unités de comparaison et travaille avec une distance de Levenshtein, classique ou ensembliste, entre modèles et mots dans les chaînes. Il a donc pour base la définition de ressemblance 3.5.3 ou celle 3.5.8 et les symboles en question sont ainsi soit des 'lettres' de l'alphabet Σ des monomères, soit des éléments d'une couverture C définie sur Σ. Nous avons montré en effet à la fin de la section 3.5.2.1.2 qu'il n'existe pas vraiment de différence entre des modèles qui sont des produits

d'ensembles d'une couverture et des modèles qui sont des mots. Une 'lettre' a de l'alphabet Σ peut en effet être écrite sous la forme d'un singleton $\{a\}$, et un mot comme un produit cartésien de tels singletons. Nous ne présentons donc ici que la version la plus générale de l'algorithme, sachant qu'il suffit de travailler avec une couverture identité $C = CI = \{\{a\} \mid a \in \Sigma\}$ pour traiter également le cas des modèles définis comme des mots. Les deux approches (modèles comme des mots et modèles comme des produits cartésiens des éléments d'une couverture) ont été introduits dans [Sagot *et al.*, 1995a] et [Sagot *et al.*, 1995d].

L'algorithme est décrit progressivement. Nous donnons d'abord une version où les erreurs ne sont pas permises entre modèles et mots. Dans un deuxième temps seules les substitutions sont autorisées, et nous présentons alors une version de l'algorithme utilisant la distance de Hamming. Enfin, nous introduisons la version utilisant la distance de Levenshtein correspondant aux définitions de ressemblance 3.5.3 et 3.5.8 du chapitre 3.

Une fois cela fait, nous montrons des modifications de l'algorithme permettant de se débarrasser en partie du phénomène de la 'multiplication' d'une position dans les ensembles d'occurrences dont nous avons parlé dans la section 5.3.1.2. Elles impliquent une perte partielle d'information durant le déroulement de l'algorithme, information qui peut cependant être facilement récupérée à la fin.

Nous terminons cette section en introduisant deux variantes de l'algorithme principal où des contraintes supplémentaires sont imposées au niveau de la définition de ressemblance entre modèles et mots ou à celui caractérisant la validité d'un modèle. Ces variantes peuvent être utiles dans certaines applications, et ont pour effet d'accélérer les calculs en pratique.

5.4.2.1.2 Algorithme sans erreurs

5.4.2.1.2.1 Définition de récurrence

Avant de donner le lemme sur lequel l'algorithme s'appuie, nous introduisons une notation simplifiée pour les EL-occurrences d'un modèle lorsque $e = 0$.

Notation 5.4.1 *Soit M un modèle défini comme un produit d'ensembles d'une couverture C. Si (i, d_s, d_d, d_i) est une EL-occurrence de M dans s telle que $d_s = d_d = d_i = 0$, alors nous écrivons simplement : $i \in EL(M)$.*

Soit s une chaîne de longueur n. Soient M, M_1, M_2 trois modèles tels que $M = M_1M_2$ et $i \in \{1,...,n\text{-}|M|+1\}$ une position dans s. Nous avons le lemme suivant :

Lemme 5.4.1

$$i \in EL(M = M_1M_2)$$

$$\Longleftrightarrow$$

$$i \in EL(M_1), (i+ \mid M_1 \mid) \in EL(M_2).$$

De ce lemme nous pouvons facilement déduire une proposition permettant de construire $EL(M)$ à partir de $EL(M_1)$ et $EL(M_2)$. Avant de la présenter, nous introduisons la notation suivante :

Notation 5.4.2 *Soit $EL(M)$ un ensemble de EL-occurrences, nous notons :*

$$(EL(M))_{-b} = \{i \mid i + b \in EL(M)\}.$$

Il s'agit tout simplement de l'ensemble obtenu de $EL(M)$ en soustrayant b de chaque élément i dans $EL(M)$.

Proposition 5.4.1 $EL(M) = EL(M_1) \cap (EL(M_2))_{-|M_1|}$.

Les preuves du lemme et de la proposition sont immédiates et sont donc omises. Ces résultats nous permettent ainsi de construire les modèles EL-présents au moins q fois dans s par longueurs croissantes.

5.4.2.1.2.2 Algorithme

Une idée de l'algorithme pour résoudre le Problème 1 est donné dans la figure 5.13. Cet algorithme est appelé Poivre (ou Moivre dans sa version pour l'ADN/ARN utilisant une couverture unitaire). Il est implémenté de manière récursive. Comme l'arbre des modèles est parcouru en profondeur, la valeur de b dans la notation 5.4.2 est égale à 1 et les modèles M_2 de la proposition 5.4.1 sont les ensembles de la couverture C de Σ. Ce sont les modèles de longueur 1.

La résolution du Problème 2 est effectuée exactement de la même façon excepté que la condition d'arrêt de la descente le long d'une branche dépend maintenant exclusivement de la contrainte de quorum q. Dans l'implémentation, la seule chose qui change est que nous devons réaliser au tout début de la fonction Explore le test supplémentaire suivant :

```
si (l > k) {
    /* un plus long modèle vient d'être rencontré */
    rejeter les modèles stockés au préalable;
    k = l;
}.
```

Observons que cet algorithme produit des résultats très similaires à ceux obtenus par KMRC (section 4.2.3.4), ce qui n'est pas étonnant dans la mesure où les définitions de ressemblance utilisées par ces algorithmes (définitions 3.4.10 et 3.5.8) sont très proches (voir section 3.5.4). Les principales différences entre les deux portent sur le fait que KMRC fait un usage uniquement implicite des modèles et d'une couverture, que cette 'couverture implicite' doit être composée de cliques maximales (aucun sous-ensemble ne peut être contenu dans un autre), et finalement que KMRC doit effectuer un test de plus à la fin de chaque étape sur les cliques d'occurrences obtenues, ceci afin d'éliminer celles qui ne sont pas maximales (section 4.2.3.4). KMRC est par ailleurs implémenté 'à la façon KMR', la longueur des occurrences est donc doublée à chaque fois. L'algorithme doit de toutes façons parcourir l'arbre en largeur s'il veut pouvoir réaliser ces tests d'inclusions sur les cliques.

5.4.2.1.2.3 Complexité

Nous avons dans ce cas $N_{Modi}(k-1) = g^{k-1}$ et $F_{MultiPos} = 1$. Comme les erreurs, et en particulier les trous, ne sont pas permises, chaque occurrence ne peut glisser qu'une seule fois le long d'une branche, elle peut par contre engendrer g nouvelles occurrences, c'est-à-dire, $N_{MaxBranches} = g$. La complexité en temps de l'algorithme est donc $O(n.N.k.g^k)$.

La complexité en espace est $O(n.N.k)$ pour un parcours en profondeur (elle serait en $O(n.N.k.g^k)$ pour un parcours en largeur).

Les valeurs pour ces complexités correspondent bien sûr à un pire cas qui ne serait observé que s'il existait un symbole de l'alphabet appartenant à tous les sous-ensembles de la couverture et si, de plus, la chaîne s était composée uniquement de ce symbole. Une telle situation est bien sûr totalement irréaliste. D'une façon plus générale, nous pouvons en fait considérer que la

```
/* Entrée */
        s = s1s2...sN : une chaîne de longueur n = concaténation des chaînes s1, s2, ..., sN
        S_1, S_2, ..., S_p : une couverture C de Σ
        q : contrainte de quorum
        k : une longueur fixe
/* Sortie */
        Tous les modèles M de longueur k qui possèdent des instances dans au moins q
        des chaînes s1, s2, ..., sN
/* Principales structures de données */
        M = produit d'ensembles de C : implémenté comme un tableau d'entier entre 1
        et p
        EL(M) = ensemble d'occurrences d'un modèle M : implémenté comme une pile
        Modèles[i] = ensembles de C auxquels S_i appartient : implémenté comme une pile
        (cette structure n'est pas vraiment nécessaire, elle est donnée afin de faciliter la
        compréhension de l'algorithme)
        ExtensionPossible = ensemble des modèles qui peuvent être étendus : implémenté
        comme une pile
Main {
        pour (i ∈ [1..n])
                pour (j ∈ [1..p]) {
                        si (i ∈ S_j)
                                EL(S_j) = EL(S_j) ∪ {i};
                        Modèles[i] = Modèles[i] ∪ S_j;
                }
        pour (j ∈ [1..p])
                si (EL(S_j) vérifie quorum de q)
                        Explore(S_j, 1, EL(S_j));
}
Explore(M, l, EL(M)) {
        si (l = k)                                      /* fin de la récursion */
                stocke M et EL(M);
        sinon {                                         /* pas de la récursion */
                pour (i ∈ EL(M))
                        pour (M_1 ∈ Modèles[i + l]) {
                                EL(MM_1) = EL(MM_1) ∪ {i};
                                ExtensionPossible = ExtensionPossible ∪ {M_1};
                        }
                pour (M_1 ∈ ExtensionPossible)
                        si (EL(MM_1) vérifie quorum de q)
                                Explore(MM_1, l + 1, EL(MM_1));
        }
}
```

Figure 5.13: Algorithme Poivre/Moivre (couverture (éventuellement unitaire) et pas d'erreurs) pour résoudre le Problème 1.

valeur de g qui est réellement importante dans le calcul de la complexité est plutôt sa valeur moyenne $\overline{g}$ telle que celle-ci est donnée dans la définition 3.2.15, à savoir :

$$\overline{g} = \frac{\sum_a g_a}{\mid \Sigma \mid}$$

où g_a est le nombre d'ensembles de C auxquels le symbole a de Σ appartient, ou bien celle donnée par la formule :

$$\overline{g} = \frac{\sum_a p_a g_a}{\mid \Sigma \mid}$$

où p_a est la probabilité d'apparition du symbole a dans la chaîne s.

5.4.2.1.3 Distance de Hamming

5.4.2.1.3.1 Définition de récurrence

Avant de donner le lemme sur lequel repose l'algorithme permettant de traiter les substitutions entre modèles et mots dans les chaînes, nous introduisons une autre notation simplifiée pour les EL-occurrences d'un modèle lorsque $e > 0$ mais que les trous ne sont pas permis.

Notation 5.4.3 *Soit M un modèle défini comme un produit d'ensembles d'une couverture. Si (i, d_s, d_d, d_i) est une EL-occurrence de M dans s telle que $d_d = d_i = 0$, alors nous écrivons $(i, d_s) \in EL(M)$.*

Soit s une chaîne de longueur n. Soient M, M_1, M_2 trois modèles tels que $M = M_1 M_2$ et $i \in \{1,...,n\text{-}|M|+1\}$ une position dans s. Alors le lemme suivant est vérifié :

Lemme 5.4.2

$$(i, d_s) \in EL(M = M_1 M_2)$$

$$\Longleftrightarrow$$

$$(i, d_{s_1}) \in EL(M_1), (i+ \mid M_1 \mid, d_{s_2}) \in EL(M_2)$$

$$et\ d_s = d_{s_1} + d_{s_2} \leq e$$

La proposition permettant de construire $EL(M)$ à partir de $EL(M_1)$ et $EL(M_2)$ est semblable à celle de la version sans erreurs, elle porte simplement sur des ensembles d'occurrences décrits différemment. Notons d'abord :

Notation 5.4.4 *Soit $EL(M)$ un ensemble de EL-occurrences, nous notons :*

$$(EL(M))_{-b} = \{(i, d_s) \mid (i + b, d_s) \in EL(M)\}.$$

Proposition 5.4.2 $EL(M) = EL(M_1) \cap (EL(M_2))_{-|M_1|}$.

5.4.2.1.3.2 Algorithme

Une idée de l'algorithme pour la résolution du Problème 1 est donné dans la figure 5.14. Cette version est appelée HPoivre (ou HMoivre dans sa version pour l'ADN/ARN).

La résolution du Problème 2 est effectuée de la même façon que pour l'algorithme sans erreur.

```
/* Entrée */
        e : le nombre maximal de substitutions autorisées
        Comme pour l'algorithme de la figure 5.13
/* Sortie */
        Comme pour l'algorithme de la figure 5.13
/* Principales structures de données */
        Modèles[i] = ensembles de la couverture C auxquels l'occurrence i appartient, avec
        le nombre de substitutions entre l'occurrence et le modèle : implémenté comme
        une pile (cette structure n'est pas vraiment nécessaire)
        Le reste comme pour l'algorithme de la figure 5.13
Main {
        pour (i ∈ [1..n])
                pour( j ∈ [1..p])
                        si (i ∈ S_j) {
                                EL(S_j) = EL(S_j) ∪ {(i, 0)};
                                Modèles[i] = Modèles[i] ∪ (S_j, 0);
                        }
                        sinon {
                                EL(S_j) = EL(S_j) ∪ {(i, 1)};
                                Modèles[i] = Modèles[i] ∪ (S_j, 1);
                        }
        pour (j ∈ [1..p])
                si (EL(j) vérifie le quorum q)
                        Explore(S_j, 1, EL(S_j));
}
Explore(M, l, EL(M)) {
        si (l = k)                                      /* base de la récursion */
                stocke M et EL(M);
        sinon {                                         /* pas de la récursion */
                pour ((i, d_s) ∈ EL(M))
                        pour ((M_1, d'_s) ∈ Modèles[i])
                                si (d'_s + d_s ≤ e) {
                                        EL(MM_1) = EL(MM_1) ∪ {(i, d_s + d'_s)};
                                        ExtensionPossible = ExtensionPossible ∪ {M_1};
                                }
                pour (M_1 ∈ ExtensionPossible)
                        si (EL(MM_1) vérifie le quorum q)
                                Explore(MM_1, l + | M_1 |, EL(MM_1));
        }
}
```

Figure 5.14: Algorithme HPoivre/HMoivre (couverture et distance de Hamming) pour résoudre le Problème 1.

5.4.2.1.3.3 Complexité

Soit i une position dans les chaînes. Calculons d'abord $N_{Modi}(k-1)$.

Soit u une EL-image correspondant à (i, d_s). En notant p le nombre d'ensembles dans la couverture C de Σ, nous avons $N_{Modi}(k-1) \leq C_1 \times C_2$ où:

- $C_1 = g^{k-1}$ est le nombre de produits d'ensembles de C auxquels un mot dans une chaîne peut appartenir (correspondant au nombre maximal de positions sans erreur de l'occurrence);
- $C_2 = \begin{pmatrix} k-1 \\ d_s \end{pmatrix}.(p-1)^{d_s}$ compte le nombre de substitutions.

Observons que le facteur C_1 surévalue le nombre de positions sans erreurs (il correspond en fait à $e = 0$). De même, le facteur C_2 peut être surévalué lorsque $g > 1$ car dans ce cas un symbole de u peut appartenir à plus d'un ensemble de la couverture.

En simplifiant et en majorant $N_{Modi}(k-1)$, nous obtenons :

$$N_{Modi}(k-1) \leq \frac{(k-1)!}{(k-1-d_s)!}.p^e.g^{k-1} \leq k^{d_s}.p^e.g^{k-1}$$

et ainsi :

$$N_{Modi}(k-1) \leq k^e.p^e.g^{k-1}.$$

$F_{MultiPos}$ est ici égal à 1 puisque les erreurs sont autorisées mais pas les trous.

Le facteur $N_{MaxBranches}$ est quant à lui majoré par $p \times F_{MultiBranche}$. Et ce dernier facteur est à nouveau égal à 1 (trous interdits). Ainsi $N_{MaxBranches} \leq p$.

La complexité en temps de l'algorithme est donc $O(n.N.k^{e+1}.p^{e+1}.g^k)$. Ce nombre correspond également au nombre maximal de nœuds effectivement présents au niveau k de l'arbre des modèles valides.

La complexité en espace est bornée supérieurement par $n.N.k$ comme dans le cas sans erreur et pour un parcours en profondeur. Elle serait cependant cette fois en $O(n.N.k^e.p^e.g^k)$ pour un parcours en largeur. Le gain ici est plus grand.

Une situation de pire cas pour cet algorithme est observée dans les mêmes circonstances que pour la version sans erreurs (chaîne s composée uniquement d'un symbole appartenant à tous les sous-ensembles de C).

5.4.2.1.4 Distance de Levenshtein

5.4.2.1.4.1 Principales différences avec la version sans erreurs ou distance de Hamming

Nous présentons maintenant le cas le plus général où au plus $e > 0$ erreurs (substitutions (non conservatives) et trous) sont autorisées entre un modèle et ses EL-occurrences. Les principales différences avec la version sans erreurs ou la version 'distance de Hamming' sont les suivantes :

1. Étant donné un modèle M, une position i d'une occurrence de M dans s peut être associée à plusieurs triplets (d_s, d_d, d_i). Chacune correspond à une image u (dont certaines peuvent être identiques) et indique une combinaison différente de substitutions, délétions et insertions permettant de passer de u à une instance de M en un nombre minimal

d'opérations (rappelons que, par convention, d_d et d_i sont le nombre de délétions et insertions réalisées sur u — voir la figure 3.25 du chapitre 3);

2. Étant donné un modèle M de longueur k et ses EL-occurrences, la longueur des EL-images correspondantes peut varier entre $\mid M \mid$ - e et $\mid M \mid + e$. Aussi, lorsque nous considérons l'occurrence (i, d_s, d_d, d_i) d'un modèle M de longueur k et que nous essayons d'étendre le modèle auquel il appartient, nous ne pouvons nous contenter de regarder la position $i+k$ pour voir quel est le prochain symbole dans s comme nous l'avions fait dans le cas sans erreurs ou avec seulement des substitutions. Nous devons maintenant considérer le fait que l'image u correspondant à l'occurrence (i, d_s, d_d, d_i) vérifie : $\mid u \mid = k$ - *déc*, où $déc = d_d - d_i$;

3. Lorsque nous juxtaposons deux EL-images u et u' des modèles M and M' respectivement, nous savons que $distance_{EL}(uu', MM') \leq distance_{EL}(u, M) + distance_{EL}(u', M') \leq e$. Nous n'avons donc pas toujours une stricte égalité. En termes de programmation dynamique, cela signifie que la concaténation de deux chemins optimaux ne produit pas toujours un chemin optimal. Ce que nous pouvons garantir, c'est qu'au moins une des concaténations correspond à un chemin optimal (le lemme 5.4.2.1.4.2 présenté un peu plus loin nous garantit cela). Ainsi uu' est une EL-image de MM'. La $distance_{EL}$ de uu' à une plus proche instance de MM' est la plus petite parmi toutes celles qui sont obtenues et est celle de Levenshtein.

Ces considérations conduisent à un algorithme différent, plus complexe que celui donné précédemment.

5.4.2.1.4.2 Définitions de récurrence

Comme auparavant, soit s une chaîne de longueur n. Soient également M, M_1, M_2 trois modèles tels que $M = M_1M_2$ et $i \in \{1,...,n\text{-}|M|+1\}$ une position dans s. Le lemme 5.4.2 doit maintenant être écrit en deux parties qui ne sont plus symétriques :

Lemme 5.4.3

$$(i, d_s, d_d, d_i) \in EL(M)$$
$$\Longrightarrow$$
$$\exists\, d_{s_1}, d_{d_1}, d_{i_1}, d_{s_2}, d_{d_2}, d_{i_2} \textit{ tel que}$$
$$(i, d_{s_1}, d_{d_1}, d_{i_1}) \in EL(M_1), (i+ \mid M_1 \mid -d_{d_1} + d_{i_1}, d_{s_2}, d_{d_2}, d_{i_2}) \in EL(M_2)$$
$$\textit{et } d_s = d_{s_1} + d_{s_2},\ d_d = d_{d_1} + d_{d_2},\ d_i = d_{i_1} + d_{i_2}.$$

Lemme 5.4.4

$$(i, d_{s_1}, d_{d_1}, d_{i_1}) \in EL(M_1),\ (i+ \mid M_1 \mid -d_{d_1} + d_{i_1}, d_{s_2}, d_{d_2}, d_{i_2}) \in EL(M_2)$$
$$\textit{et } d_{s_1} + d_{d_1} + d_{i_1} + d_{s_2} + d_{d_2} + d_{i_2} \leq e$$
$$\Longrightarrow$$
$$(i, d_{s_1} + d_{s_2}, d_{d_1} + d_{d_2}, d_{i_1} + d_{i_2}) \in ES(M_1M_2 = M) \supseteq EL(M)$$
$$\textit{où } ES(M) = \{(i, d_s, d_d, d_i) \mid d_s + d_d + d_i \leq e\}.$$

Avant de présenter la propriété permettant dans ce cas de construire $EL(M)$ à partir de $EL(M_1)$ et $EL(M_2)$, nous introduisons une notation, une opération de filtrage et une opération spéciale d'intersection sur les ensembles de EL-occurrences des deux modèles M_1 et M_2.

Notation 5.4.5 *Soit $EL(M)$ un ensemble de EL-occurrences. Nous notons :*

$$(EL(M))_{-b} = \{(i,d_s,d_d,d_i) \mid (i+b,d_s,d_d,d_i) \in EL(M)\}.$$

Définition 5.4.1 *Soit un ensemble $ES(M) = \{(i, d_s, d_d, d_i) \mid d_s+d_d+d_i \leq e\}$. Nous définissons l'opération $Filtre(ES(M))$ par :*

$$Filtre(ES(M)) = \{(i, d_s, d_d, d_i) \in ES(M) \mid d = d_s + d_d + d_i \text{ est une } distance_{EL}\}.$$

En d'autres termes :

$$Filtre(ES(M)) = \{(i, d_s, d_d, d_i) \in ES(M) \mid d_s + d_d + d_i \leq d'_s + d'_d + d'_i,\ \forall (i, d'_s, d'_d, d'_i) \in ES(M) \text{ avec } d'_d - d'_i = d_d - d_i\}.$$

La dernière condition exprime simplement le fait que deux occurrences doivent avoir même longueur, c'est-à-dire, $\mid M \mid - d_d + d_i = \mid M \mid - d'_d + d'_i$.

Définition 5.4.2 *Étant donné deux ensembles de EL-occurrences $EL(M_1)$ et $EL(M_2)$, nous définissons l'opération $\sqcap_{EL}$ par :*

$$\begin{aligned} EL(M_1) \sqcap_{EL} EL(M_2) = \{&(i,d_s,d_d,d_i) \mid (i,d_{s_1},d_{d_1},d_{i_1}) \in EL(M_1), \\ &(i + d_{d_1} - d_{i_1},d_{s_2},d_{d_2},d_{i_2}) \in EL(M_2), \\ &d_s = d_{s_1} + d_{s_2},\ d_d = d_{d_1} + d_{d_2},\ d_i = d_{i_1} + d_{i_2} \\ &\text{et } d_s + d_d + d_i \leq e\}. \end{aligned}$$

Des lemmes il vient alors :

Proposition 5.4.3 $EL(M) = Filtre(EL(M_1) \sqcap_{EL} (EL(M_2))_{-|M_1|})$.

5.4.2.1.4.3 Algorithme

Une idée générale de l'algorithme pour résoudre le Problème 1 est donnée dans la figure 5.15. Cet algorithme est appelé LePoivre (LeMoivre si $C = CI$). La modification qui doit être réalisée afin de traiter le Problème 2 est semblable à celle indiquée pour l'algorithme Poivre/Moivre.

5.4.2.1.4.4 Complexité

Soit i une position dans les chaînes. Étant donné trois valeurs d_s, d_d et d_i telles que $d_s + d_d + d_i \leq e$, nous calculons d'abord $N_{Modi}(k-1)$.

Soit u une EL-image correspondant à (i, d_s, d_d, d_i). En notant $déc = d_d - d_i$ et p le nombre d'ensembles dans la couverture C de Σ, nous avons $N_{Modi}(k-1) \leq C_1 \times C_2 \times C_3 \times C_4$ où:

- $C_1 = g^{k-1}$ est le nombre de produits d'ensembles de C auxquels un mot dans une chaîne peut appartenir (correspondant au nombre maximal de positions sans erreur de l'occurrence);

```
/* Entrée */
        e : le nombre maximal d'erreurs autorisées
        Le reste comme pour l'algorithme de la figure 5.13
/* Sortie */
        Comme pour l'algorithme de la figure 5.13
/* Principales structures de données */
        Modèles[i] = ensembles de la couverture C auxquels l'occurrence i appartient, avec
        le nombre de substitutions, délétions et insertions entre l'occurrence et l'instance
        la plus proche de l'ensemble : implémenté comme une pile
        (à nouveau ici, cette structure n'est pas vraiment nécessaire, les modèles de longueur
        1 pouvant être 'reconstruits' à chaque fois — elle est donnée pour faciliter)
        Le reste comme pour l'algorithme de la figure 5.13
Main {
        pour (i ∈ [1..n])
                pour (j ∈ [1..p])
                        si (i ∈ S_j) {
                                EL(S_j) = EL(S_j) ∪ {(i, 0, 0, 0)};
                                Modèles[i] = Modèles[i] ∪ (S_j, 0, 0, 0);
                        }
                        sinon {
                                EL(S_j) = EL(S_j) ∪ {(i, 1, 0, 0)} ∪ {(i, 0, 1, 0)}};
                                Modèles[i] = Modèles[i] ∪ (S_j, 1, 0, 0) ∪ (S_j, 0, 1, 0) ∪ (∅, 0, 0, 1);
                        }
        pour (j ∈ [1..p])
                si (EL(j) vérifie le quorum q)
                        Explore(S_j, 1, EL(S_j));
}
Explore(M, l, EL(M)) {
        si (l = k)                                      /* base de la récursion */
                stocke M et EL(M);
        sinon {                                         /* pas de la récursion */
                pour ((i, d_s, d_d, d_i) ∈ EL(M))
                        pour ((M_1, d'_s, d'_d, d'_i) ∈ Modèles[i + l − d_d + d_i])
                                si (d'_s + d'_d + d'_i + d_s + d_d + d_i ≤ e) {
                                        ES(MM_1) = ES(MM_1) ∪ {(i, d_s + d'_s, d_d + d'_d, d_i + d'_i)};
                                        ExtensionPossible = ExtensionPossible ∪ {M_1};
                                }
                pour (M_1 ∈ ExtensionPossible)
                        si (EL(MM_1) = Filtre(ES(MM_1)) vérifie le quorum q)
                                Explore(MM_1, l + | M_1 |, EL(MM_1));
        }
}
```

Figure 5.15: Algorithme LePoivre/LeMoivre (couverture et distance de Levenshtein) pour résoudre le Problème 1.

- $C_2 = \begin{pmatrix} k-1-déc \\ d_s \end{pmatrix}.(p-1)^{d_s}$ compte le nombre de substitutions;
- $C_3 = \begin{pmatrix} k-1-déc-d_s \\ d_i \end{pmatrix}$ compte le nombre d'insertions une fois que les substitutions ont été considérées;
- $C_4 = A^{d_d}_{k-1-déc}.p^{d_d}$ compte le nombre de délétions avec $A^{d_d}_{k-1-déc} = \frac{(k-1-déc)!}{(k-1-déc-d_d)!}$ (comme il s'agit juste de choisir des emplacements pour les délétions, ces évènements ne peuvent être distingués).

Comme précédemment, les facteurs C_1 et C_2 peuvent être surévalués.

En simplifiant et en majorant $N_{Modi}(k-1)$, nous obtenons que $N_{Modi}(k-1)$ est au plus égal à :

$$\frac{(k-1-déc)!(k-1-déc-d_s)!(k-1-déc)!}{(k-1-déc-d_s)!(k-1-déc-d_s-d_i)!(k-1-déc-d_d)!}.p^e.g^{k-1}.$$

Nous avons donc que :

$$N_{Modi}(k-1) \leq k^{d_s+d_i+d_d}.p^e.g^{k-1}$$

et ainsi :

$$N_{Modi}(k-1) \leq k^e.p^e.g^{k-1}.$$

Il nous faut maintenant calculer la valeur de $F_{MultiPos}$. Cela revient à calculer le nombre de triplets possibles (d_s, d_d, d_i) pour une position donnée i.

Soit u une EL-image d'un modèle M commençant à la position i. La longueur de u peut varier entre $\mid M \mid$ - e et $\mid M \mid$ + e uniquement, un total de $2e+1$ possibilités. Notons e' la distance Ensembliste de Levenshtein entre l'image et son modèle. Quel que soit le triplet (d_s, d_d, d_i) associé à u, nous devons avoir $d_s + d_d + d_i = e'$ et $d_d - d_i = \mid M \mid - \mid u \mid = c$ où c est une constante. Aussi $d_d = e' + c - \frac{d_s}{2}$ et il ne peut y avoir plus de $\lfloor \frac{e+1}{2} \rfloor$ tels triplets associés à u. Il ne peut y avoir donc plus de $\lfloor \frac{e+1}{2} \rfloor.(2e+1)$ éléments (d_s, d_d, d_i) pour lesquels $d_s + d_d + d_i$ représente une distance Ensembliste de Levenshtein entre u et un modèle M. Si toutefois nous désirons uniquement localiser les modèles et leurs occurrences et non pas aligner ces objets les uns par rapport aux autres exactement (ce qui est notre cas), alors deux occurrences situées à une même position i dans s, ayant même longueur et présentant le même nombre minimal total d'erreurs avec un modèle peuvent être considérées comme étant équivalentes bien qu'elles puissent différer individuellement dans le nombre de substitutions, délétions et insertions qu'elles présentent par rapport au modèle. Nous n'avons donc à considérer, pour chaque position i, que $(2e+1)$ occurrences possibles. Ainsi $F_{MultiPos} \leq 2e+1$.

Comme les trous sont cette fois-ci autorisés, le facteur $N_{MaxBranches}$ est majoré par $p \times F_{MultiBranche}$. Et ce dernier facteur est lui-même borné supérieurement par le nombre maximal d'insertions permises (égal au nombre maximal d'erreurs) plus deux (l'occurrence d'un modèle M de longueur $k-1$ n'est pas étendue (délétion) ou est prolongée sans trous par rapport à un modèle de longueur k ayant M comme préfixe). Ainsi $N_{MaxBranches} \leq (e+2).p$.

La complexité en temps de l'algorithme (opération de filtrage exclue) est donc $O(n.N.(e+2).(2e+1).k^{e+1}.p^{e+1}.g^k)$ (observons que, au contraire de ce qui se passait pour la version sans erreurs ou distance de Hamming, le nombre maximal de nœuds effectivement présents au niveau k de l'arbre des modèles valides peut être inférieur au nombre maximal d'opérations nécessaires pour y parvenir).

Finalement, si les éléments de $EL(M)$ sont maintenus en permanence ordonnés (cet ordre découle naturellement de l'implémentation de l'algorithme), l'opération de filtrage peut être

réalisée en utilisant un tableau d'entiers de taille $(2e+1)$ où les meilleurs scores rencontrés pour chaque position i et longueur $\mid u \mid$ sont préservés. L'usage d'une pile additionnelle indiquant quelles cellules de ce tableau ont été modifiées pour chaque position i nous permet de ne l'initialiser qu'une seule fois au début de l'algorithme (avant l'entrée dans la fonction récursive Explore). Le filtrage peut ainsi être effectué en temps constant.

La complexité en temps de l'algorithme demeure donc en $O(n.N.(e+2).(2e+1).k^{e+1}.p^{e+1}.g^k)$.

Sa complexité en espace est bornée supérieurement par $n.N.k.F_{MultiPos}$ et est donc en $O(n.N.k.(2e+1))$ pour un parcours en profondeur (elle serait en $O(n.N.(2e+1).k^e.p^e.g^k)$ pour un parcours en largeur).

5.4.2.1.5 Perte d'information récupérable par la suite

5.4.2.1.5.1 Introduction

L'idée derrière les modifications de l'algorithme précédent qui sont présentées dans cette section consiste à accepter de perdre un peu d'information en cours de route de façon à la fois à essayer d'économiser en espace et à aller plus vite, sachant que cette information perdue n'est pas essentielle pour le déroulement de l'algorithme (tous les modèles valides sont quand même repérés) et qu'elle peut être facilement récupérée ensuite. En fait, la seule information qu'il est absolument nécessaire de préserver est celle concernant les modèles en tant que tels, c'est-à-dire leurs 'noms'. Les diverses versions données ici varient donc suivant la quantité d'information supplémentaire qu'elles maintiennent en mémoire.

5.4.2.1.5.2 Algorithmes

Essentiellement trois modifications sont possibles :

1. la première consiste précisément à ne garder que les 'noms' des modèles valides, aucune information concernant les occurrences de ces modèles n'étant conservée;
2. une seconde version conserve non seulement l'indication des modèles valides mais également, pour chacun d'eux, celle de la position de la première occurrence de ces modèles rencontrée sur chacune des chaînes;
3. la dernière enfin garde tous les noms des modèles qui sont valides, ainsi que toutes les positions où ces modèles sont présents dans les chaînes, mais ne distingue plus les occurrences en termes de quadruplets (i, d_s, d_d, d_i).

Concernant la première modification, il convient d'observer qu'elle n'est pas équivalente à l'approche naïve qui consiste à engendrer tous les modèles d'une certaine longueur car elle tient compte du quorum. Les modèles sont ainsi toujours construits par longueurs croissantes et un modèle M est invalidé, de même que tous les modèles ayant M pour préfixe, si M ne vérifie plus la contrainte de quorum. L'algorithme est alors très simple puisqu'il se réduit à une recherche avec erreur d'un motif dans un texte. Comme la longueur des modèles trouvés sur des exemples biologiques ne dépasse pas en général la taille d'un mot machine (32 bits), nous avons choisi d'utiliser pour cette recherche l'algorithme Agrep de Wu et Manber [Wu and Manber, 1992a] [Wu and Manber, 1992b] qui lui-même est une version étendue de l'algorithme 'Shift-Add' de Baeza-Yates et Gonnet [Baeza-Yates and Gonnet, 1992]. La recherche d'un motif de longueur k dans un texte de longueur n avec au plus e erreurs se fait en $O(n.e)$. Elle est ainsi indépendante de la longueur du motif (si celui-ci est plus petit qu'un mot machine).

La seconde modification utilise le même algorithme que précédemment, simplement cette fois, pour chaque modèle M, sont conservées les positions de début de la première occurrence de M sur chacune des chaînes (au maximum N positions sont ainsi préservées). Lorsqu'un modèle M est allongé en un modèle MS avec $S \in C$, la recherche d'une première occurrence de MS sur chacune des chaînes peut se faire alors en toute sécurité non plus à partir du début des chaînes, mais à partir de la position précédemment obtenue sur chacune d'elles.

La troisième version est plus complexe et ne fait pas appel à des algorithmes de recherche d'un motif dans un texte. Sa validité repose sur l'observation suivante. Supposons qu'un modèle M soit présent de manière multiple à la position i d'une chaîne s, c'est-à-dire, plusieurs occurrences du modèle ayant des longueurs différentes se trouvent à la position i avec un nombre d'erreurs pouvant varier entre 0 et e. Si M est allongé en un modèle MS, il n'est pas possible à ce stade de savoir lesquelles parmi les occurrences de M localisées à la position i vont pouvoir être prolongées. Il faut donc toutes les conserver. Mais considérons maintenant le problème d'une autre façon. Supposons que cette fois ce qui est préservé est la position de fin des occurrences de M et non celle de début. Chaque position de s peut ainsi à nouveau être associée à plusieurs occurrences de M. Pour savoir si la position $i+1$ est la fin d'une occurrence du modèle MS obtenu par allongement de M, il suffit de vérifier que la meilleure occurrence de M (celle présentant le plus petit nombre d'erreurs) dont la fin se situe à la position i peut être prolongée. En effet, si cela n'était pas le cas, aucune des autres occurrences se terminant en i ne pourrait l'être non plus. Pour chaque modèle M, il suffit donc de conserver l'ensemble des positions i indiquant la fin d'au moins une occurrence de M, ainsi que la distance de la meilleure occurrence se terminant en i. Une position i ne peut ainsi appartenir qu'une seule fois à l'ensemble $EL(M)$ des occurrences de M et non plus $2e+1$ fois comme auparavant. Au niveau de l'implémentation, une précaution est toutefois nécessaire dans la mesure où les positions de fin des occurrences d'un modèle allongé MS peuvent ne pas être produites de manière ordonnée (voir la figure 5.16). Afin de maintenir un tel ordre dans les listes d'occurrences des modèles et ne pas y introduire deux fois une même position, il faut donc, à chaque fois que nous voulons ranger une nouvelle position dans ces listes, examiner si elle ne s'y trouve pas déjà. Deux cas de figures sont alors possibles :

- la position — notons-la i — actuellement considérée ne se trouve pas dans $EL(MS)$. Nous sommes sûrs alors que la dernière position de la liste correspondant à $EL(MS)$ est plus petite que i et qu'il suffit donc d'ajouter i à la fin de cette liste (en pratique, en haut de la pile puisque c'est ainsi que ces listes sont implémentées). La preuve fait l'hypothèse que les ensembles d'occurrences sont maintenus en permanence ordonnés (par positions croissantes) et est par induction sur la longueur des modèles. Supposons par l'absurde qu'il existe une position $i+p$ avec $p > 0$ telle que $i+p$ indique la fin d'une occurrence de MS qui aurait été placée dans $EL(MS)$ auparavant. Comme $EL(M)$ est ordonnée (hypothèse d'induction), cela signifie qu'une occurrence de MS se terminant en $i+p$ présentait au moins $p+1 \leq e$ insertions par rapport à MS. Mais alors, l'occurrence se terminant en i présenterait p insertions de moins et aurait également été placée dans $EL(MS)$ au moment où $i+p$ l'a été (par rapport à la même occurrence de M), ce qui contredirait le fait que i ne se trouve pas dans $EL(MS)$;

- la position i se trouve déjà dans $EL(MS)$. Dans ce cas, cette position n'y est pas placée une seconde fois et seul le nombre d'erreurs associé (correspondant à la distance minimale) est éventuellement modifié.

Dans les deux situations, il faudra examiner au plus e des derniers éléments de $EL(MS)$ à chaque fois que l'on voudra y placer une position nouvelle. Dans la meilleure des hypothèses

Figure 5.16: Exemple de l'ordre suivant lequel les occurrences de longueur $k+1$ sont produites à partir de celles de longueur k dans la troisième version modifiée de LePoivre.

(qui sera fréquemment observée pour des modèles plus longs), une seule vérification suffira (la fin de l'occurrence précédemment placées dans la liste se trouvant déjà à une distance supérieure à e de celle que l'on veut y introduire).

5.4.2.1.5.3 Complexités

Pour les deux premières versions de l'algorithme modifié, les facteurs de multiplicité $F_{MultiPos}$ et $F_{MultiBranches}$ disparaissent. Les autres se trouvent multiplés par le nombre d'opérations pour rechercher un motif dans un texte, ici N chaînes de longueur moyenne n.

La première version effectue cette recherche à partir du début des chaînes à chaque fois, la complexité est donc en $O(n.N.e.n.N.k^{e+1}.p^{e+1}.g^k)$ où k est une longueur fixe ou la plus grande possible, auquel s'ajoute le temps nécessaire pour récupérer toutes les occurrences à la fin (rappelons que seul le nom des modèles valides est initialement identifié). Ce second temps dépend bien sûr du nombre de ces modèles. Notons ce nombre $N_{ModèlesValides}$. Dans le pire des cas, il est égal à $n.N.(2e+1).N_{modi}(k)$, mais en pratique il peut être bien plus petit que cela. La complexité en temps totale est donc en $O(n^2.N^2.e.k^{e+1}.p^{e+1}.g^k + k.e.N_{ModèlesValides})$. Celle en espace est en $O(k)$: il suffit de conserver le nom du modèle en construction.

La complexité en temps de la seconde version est similaire à celle de la première excepté que, comme cette fois la position de la première occurrence de chaque modèle sur chacune des chaînes est conservée, celles-ci ne sont parcourues qu'une seule fois par rapport à chaque modèle. Par contre, il faut refaire la recherche à partir du début de l'occurrence du modèle préfixe de celui en train d'être construit, ce qui signifie qu'à chaque allongement d'un modèle, $N.k.e$ comparaisons 'en trop' sont effectuées. La complexité totale en temps est donc en $O(n.N^2.k.e.k^{e+1}.p^{e+1}.g^k + k.e.N_{ModèlesValides})$. Celle en espace est en $O(N.k)$: il faut conserver le nom du modèle en construction et les positions de ses premières occurrences sur chaque chaîne.

Enfin, dans la complexité en temps de la troisième et dernière version, le terme multiplicatif des positions $F_{MultiPos}$ disparaît et est substitué par le nombre maximal de 'regards dans la liste ordonnée' qu'il faut réaliser : $e+1$. À cela s'ajoute le temps pour récupérer toutes les occurrences à la fin. Ici, toutes les positions sont connues, il suffit donc de retrouver les mots correspondants (leurs longueurs). Cela peut se faire avec un simple algorithme de programmation dynamique sur des objets de longueur k (modèle) et $k+e$ (occurrence) respectivement. La complexité en temps finale est donc en $O(n.N.(e+1).(e+2).k^{e+1}.p^{e+1}.g^k) + k.(k+e).N_{ModèlesValides})$. La complexité en espace est en $O(n.N.k)$. Cette dernière version devrait donc être à la fois un peu plus rapide que la version originale (LePoivre), tout en étant souvent bien plus économe en mémoire. Cela sera expérimentalement vérifié dans la section performance.

5.4.2.1.6 Variantes : introduction de contraintes supplémentaires

5.4.2.1.6.1 Distribution uniforme des erreurs

L'algorithme présenté ci-dessus nous permet de résoudre le problème général énoncé dans la section 5.1 par rapport aux définitions de ressemblance 3.5.3 et 3.5.8. D'un point de vue pratique cependant, ces définitions peuvent être trop larges. Ainsi, toutes les erreurs d'une occurrence par rapport à un modèle peuvent apparaître groupées au début ou à la fin des deux objets, or cela n'est pas toujours très satisfaisant du point de vue de la biologie. Nous pouvons donc être amenés à introduire des contraintes sur la façon dont sont distribuées les erreurs entre un modèle et ses occurrences. En particulier, il est aisé de modifier l'algorithme de telle sorte que cette distribution soit uniforme.

Étant donné un modèle M et une quelconque de ses images u, nous demandons dans ce cas que chaque sous-modèle de longueur $l \leq \mid M \mid$ de M possède au plus un nombre $e_l < e$ d'erreurs avec le sous-mot correspondant de u, c'est-à-dire au plus un total de e_l substitutions, délétions ou insertions avec ce sous-mot. Lorsque M est étendu en un modèle $M' = MM_1$ avec $M_1 \in C$, le mot étendu correspondant $u' = ua$ avec $a \in \Sigma$ peut donc ne plus être une image de M' si $a \notin M_1$ et si u présente e_l erreurs avec les dernières $l-1$ positions de M. En termes d'implémentation, ce qui change par rapport à l'algorithme de la section 5.4.2.1.4 ou 5.4.2.1.5 est que nous devons maintenant conserver, pour chaque image u, les positions (relativement au modèle M) des dernières e_l erreurs dans u.

Par exemple, pour $e_l = 1$, une valeur entière additionnelle doit être stockée avec chaque occurrence. Cette valeur — notons-la p_l — est incrémentée à chaque extension d'une image contre un modèle tant qu'aucune erreur n'est introduite. Lorsque une image u ne peut être étendue 'exactement' par rapport à un modèle $M' = MM_1$, nous devons vérifier que $p_l > e_l$. Si cela est le cas, et si le nombre total d'erreurs de u par rapport à M est moins que e, alors l'extension de u est une occurrence de M' et p_l est remis à 0. Cette manière de procéder restreint, surtout au début de la construction, le nombre de modèles valides.

5.4.2.1.6.2 Présence exacte des modèles au moins une fois

Il est tout à fait possible qu'un modèle soit présent au moins q fois dans les chaînes de l'ensemble sans être jamais instancié dans aucune de celles-ci (c'est-à-dire, sans jamais être présent avec 0 erreur). Dans le cadre spécifique d'une recherche sur banque, il peut être intéressant de demander qu'un modèle soit instancié au moins dans la séquence requête. Cette exigence produit une seconde variante de l'algorithme où nous demandons que chaque modèle M possède au moins une occurrence i dont l'image correspondante u appartienne à M, c'est-à-dire soit une instance de M dans s. Afin de vérifier cela, nous avons besoin uniquement de placer une variable booléenne dans le corps de la récursion de la fonction Explore qui indique si $EL(MM_1)$ contient au moins une occurrence ne présentant aucune erreur avec son modèle. Si cela est le cas, MM_1 est un modèle valide.

Cette contrainte peut considérablement réduire l'espace de recherche des modèles et, en conséquence, accélérer l'algorithme. Cette accélération sera d'autant plus grande que le nombre d'erreurs autorisé entre un modèle et ses occurrences est lui aussi grand.

Bien sûr, les deux variantes présentées ici peuvent être combinées.

5.4.2.1.7 Performance en pratique

Nous présentons ici diverses figures qui indiquent les performances pratiques de H/LeMoivre, H/LePoivre et des modifications des deux algorithmes décrites dans la section 5.4.2.1.5. Toutes ces mesures ont été effectuées sur une Sparc Station 20.

Les trois premières figures, 5.17, 5.18 et 5.19, concernent H/LeMoivre. L'ensemble de chaînes avec lequel nous avons travaillé est constitué de 950 séquences d'ADN de longueur 40 nucléotides, dont chacune comporte deux signaux, l'un très court représentant le codon de début de la traduction (en général ATG) et un autre un peu plus long (constitué de 9 à 10 bases) correspondant au site de fixation d'un complexe macromoléculaire (protéine et ARN ribosomique) appelée ribosome qui participe au processus de la traduction. La couverture utilisée dans ce cas est, bien sûr, la couverture identité CI (voir notation 3.2.2) et les modèles sont donc ici des mots sur le même alphabet que celui des chaînes. Les figures 5.17 et 5.18 donnent les courbes du temps d'exécution (en secondes) de HMoivre et LeMoivre respectivement, par

Variation erreurs	HMoivre		LeMoivre	
	Valeur théor.	Valeur moy.	Valeur théor.	Valeur moy.
$0 \rightarrow 1$	12	9,2	54	15,6
$1 \rightarrow 2$	12	3	26,7	5,6
$2 \rightarrow 3$	12	1,8	21	2,4

Tableau 5.1: Rapport (théorique et pratique) des temps d'exécution de H/LeMoivre lorsque le nombre d'erreurs varie de e à $e+1$ pour $q = 80\%$, $e = 0$, 1, 2 et k fixe.

rapport à un nombre de chaînes variant entre 50 et 950 avec un pas de 50, un quorum fixé à 80%, et pour $k = 3$ (courbes 1) ou $k = k_{max}$ (courbes 2). Ce qu'il est important d'observer dans ces séries de courbes est que :

- elles démontrent bien la linéarité de H/LeMoivre par rapport au nombre N de chaînes, et donc par rapport à la longueur totale $n.N$ de celles-ci (puisque leurs longueurs est toujours la même);
- ces courbes indiquent que le temps d'exécution de ces algorithmes croît moins lentement avec le nombre d'erreurs que la valeur théorique de la complexité ne le laisserait supposer ainsi que l'indique le tableau 5.1.

Notons que le temps d'exécution demeure linéaire avec la longueur totale $n.N$ des chaînes même lorsque des modèles de longueur maximale sont recherchés. Cette dernière varie cependant peu dans cet exemple, pour des N croissants et par rapport au même nombre d'erreurs, sans doute à cause de la présence du site de fixation du ribosome qui est relativement bien conservé partout. Cette longueur maximale est de 3, (6 ou 5), (8 ou 7) et (10 ou 9) pour HMoivre pour e égal à 0 jusqu'à 3 respectivement; elle est de 3, (6 ou 5), 8 et 10 pour LeMoivre.

Les courbes 1 de la figure 5.19 indiquent quant à elles le temps d'exécution (en secondes) de HMoivre (courbe 1a) et de LeMoivre (courbe 1b) pour un quorum variant entre 5% et 100% avec un pas de 5% lorsque les plus longs modèles sont recherchés sur 100 chaînes avec 2 substitutions ou 2 erreurs permises respectivement. Dans ce cas, les valeurs de k_{max} ont varié entre 15 et 6 pour HMoivre et entre 16 et 6 pour LeMoivre. Les courbes 2 de la même figure indiquent enfin le temps d'exécution des deux algorithmes pour k variant de 2 à 11 avec un pas de 1, $N = 10$ chaînes, $q = 50\%$ et $e = 2$.

Les trois figures suivantes, 5.20, 5.21 et 5.22 concernent H/LePoivre cette fois. Dans ce cas, le jeu d'essai a consisté en un ensemble de 38 protéines appartenant à deux sous-familles de nitrogénases, à savoir les chaînes alpha et beta de la composante 1 [Pau, 1989]. Nous travaillerons à nouveau avec ces protéines, qui forment un complexe oligomérique responsable de la fixation de l'azote, dans le chapitre des illustrations. Cette famille contient au moins deux motifs consensus (PROSITE [Bairoch, 1992] numéros PS00699 et PS0090), tous deux correspondant à des sites actifs de la protéine. La longueur de ces chaînes varie entre 441 et 533 acides aminés, avec une moyenne de 500. La couverture C utilisée est :

Figure 5.17: Performance de HMoivre : temps (en secondes) $\times$ nombre de chaînes pour $k = 3$ (courbes 1) et $k = k_{max}$ (courbes 2), $q = 80\%$ et pour 0, 1, 2 et 3 substitutions (courbes a, b, c et d respectivement).

Figure 5.18: Performance de LeMoivre : temps (en secondes) $\times$ nombre de chaînes pour $k = 3$ (courbes 1) et $k = k_{max}$ (courbes 2), $q = 80\%$ et pour 0, 1, 2 et 3 erreurs (courbes a, b, c et d respectivement).

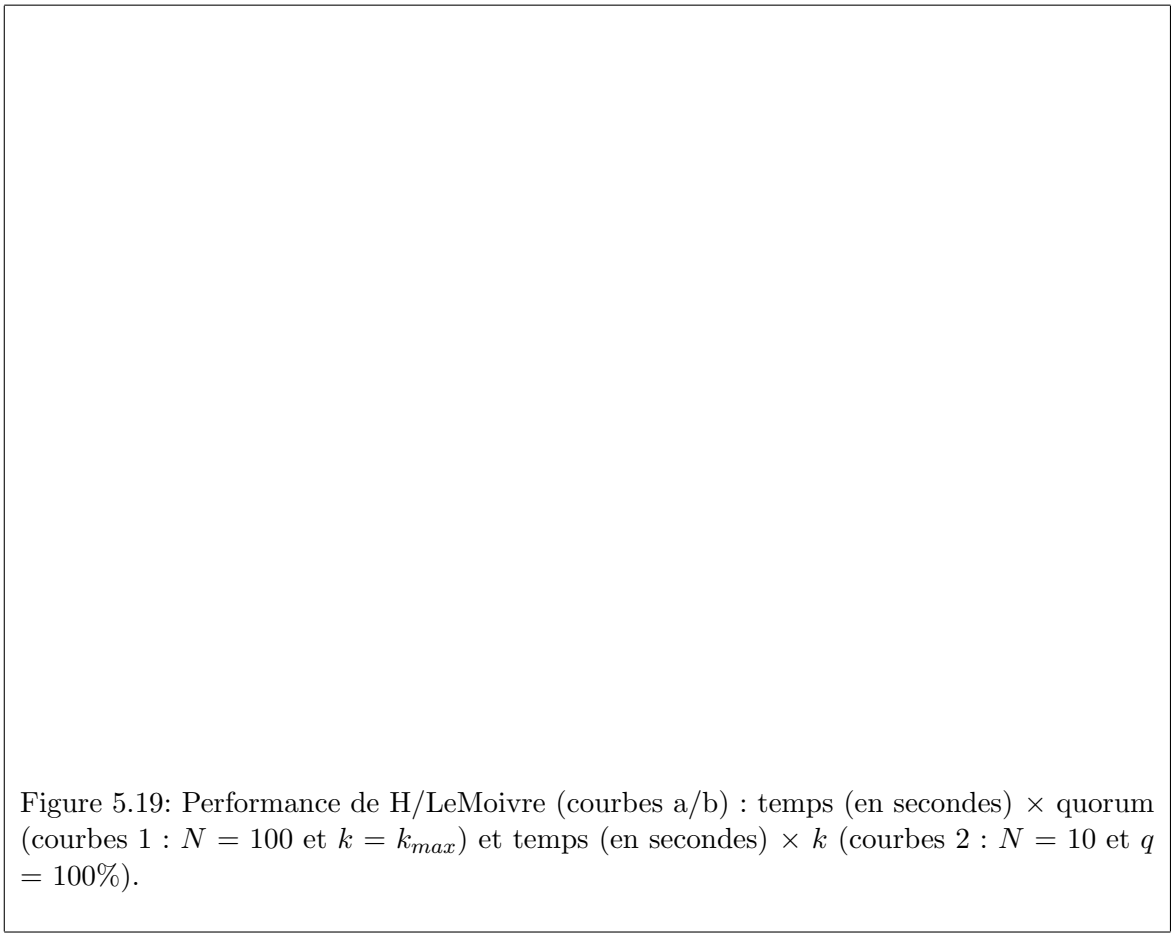


Figure 5.19: Performance de H/LeMoivre (courbes a/b) : temps (en secondes) $\times$ quorum (courbes 1 : $N = 100$ et $k = k_{max}$) et temps (en secondes) $\times$ k (courbes 2 : $N = 10$ et $q = 100\%$).

Variation erreurs	HPoivre		LePoivre	
	Valeur théor.	Valeur moy.	Valeur théor.	Valeur moy.
$0 \rightarrow 1$	40	36	180	81
$1 \rightarrow 2$	40	10	89	17

Tableau 5.2: Rapport (théorique et pratique) des temps d'exécution de H/LePoivre lorsque le nombre d'erreurs varie de e à $e+1$ pour $e = 0$ ou 1, $q = 100\%$ et pour k fixe.

$S_1 = \{A, C, G, S, T\}$ — très petit
$S_2 = \{A, C, D, G, N, S, T, V\}$ — petit
$S_3 = \{F, H, W, Y\}$ — aromatique
$S_4 = \{V, L, I, M\}$ — hydrophobe
$S_5 = \{K, R, H\}$ — basique
$S_6 = \{D, E\}$ — acide
$S_7 = \{Q, N\}$ — amide
$S_8 = \{P\}$ — proline

Elle est inspirée du diagramme de Taylor (section 3.2.3.3.1). Comme auparavant, les figures 5.20 et 5.21 indiquent la variation du temps avec le nombre de chaînes (et donc avec la longueur totale puisque les longueurs des chaînes sont approximativement les mêmes) pour HPoivre et LePoivre. Cette fois cependant, nous ne présentons les courbes que pour e allant de 0 à 2 lorsque k est fixé et pour e égal à 0 ou 1 lorsque $k = k_{max}$. Le quorum dans ce cas est égal à 100% et le nombre de chaînes varie entre 2 et 38 avec un pas de 2.

Les observations faites à propos des courbes pour H/LeMoivre demeurent valables pour H/LePoivre, et en particulier le tableau 5.2 montre que le temps d'exécution croît moins lentement avec le nombre d'erreurs que ne le laisserait supposer la complexité théorique. Les longueurs maximales observées varient cette fois de 16 à 6 et de 30 à 9 pour HPoivre et un nombre de substitutions égal à 0 ou 1 respectivement et de 16 à 6 et de 30 à 10 pour LePoivre pour les mêmes nombres d'erreurs (substitution ou trou). La différence ici est plus importante que celle observée pour H/LeMoivre sur le jeu d'essai précédent. En particulier, k_{max} décroît abruptement au départ, c'est pourquoi nous avons retiré les deux premiers points de la courbe 2b ($N = 2$, 4) pour lesquels k_{max} est égal à 30 et le temps d'exécution approximativement 3000 secondes. À partir de huit chaînes, la valeur de k_{max} demeure pratiquement constante.

Les courbes 1 de la figure 5.22 indiquent le temps d'exécution (en secondes) de HPoivre (courbe 1a) et LePoivre (courbe 1b) lorsque le quorum varie entre 50% et 100% avec un pas de 5% et que les plus longs modèles sont recherchés sur 20 chaînes avec 1 erreur permise (substitution uniquement ou substitution et trou respectivement). Les valeurs obtenues pour k_{max} se sont situées entre 16 ($q = 50\%$) et 10 ($q = 100\%$) pour les deux algorithmes. Les courbes 2 de la même figure indiquent le temps d'exécution lorsque k varie de 3 à 18 avec un pas de 1 pour 6 chaînes, $q = 100\%$ et $e = 1$.

Enfin, les trois dernières figures de cette section permettent de comparer les temps d'exécution des trois versions modifiées de LePoivre présentées dans la section 5.4.2.1.5 avec celui de LePoivre lui-même pour $q = 100\%$ et 0 (figure 5.23), 1 (figure 5.24) et 2 erreurs (figure 5.25). Le

Figure 5.20: Performance de HPoivre : temps (en secondes) $\times$ nombre de chaînes pour $k = 5$ (courbes 1) et $k = k_{max}$ (courbes 2), $q = 100\%$ et pour 0, 1 et 2 substitutions (courbes a, b et c respectivement).

Figure 5.21: Performance de LePoivre : temps (en secondes) $\times$ nombre de chaînes pour $k = 5$ (courbes 1) et $k = k_{max}$ (courbes 2), $q = 100\%$ et pour 0, 1 et 2 erreurs (courbes a, b et c respectivement).

Figure 5.22: Performance de H/LePoivre (courbes a/b) : temps (en secondes) $\times$ quorum (courbes 1 : $N = 20$ et $k = k_{max}$) et temps (en secondes) $\times$ k (courbes 2 : $N = 6$ et $q = 100\%$).

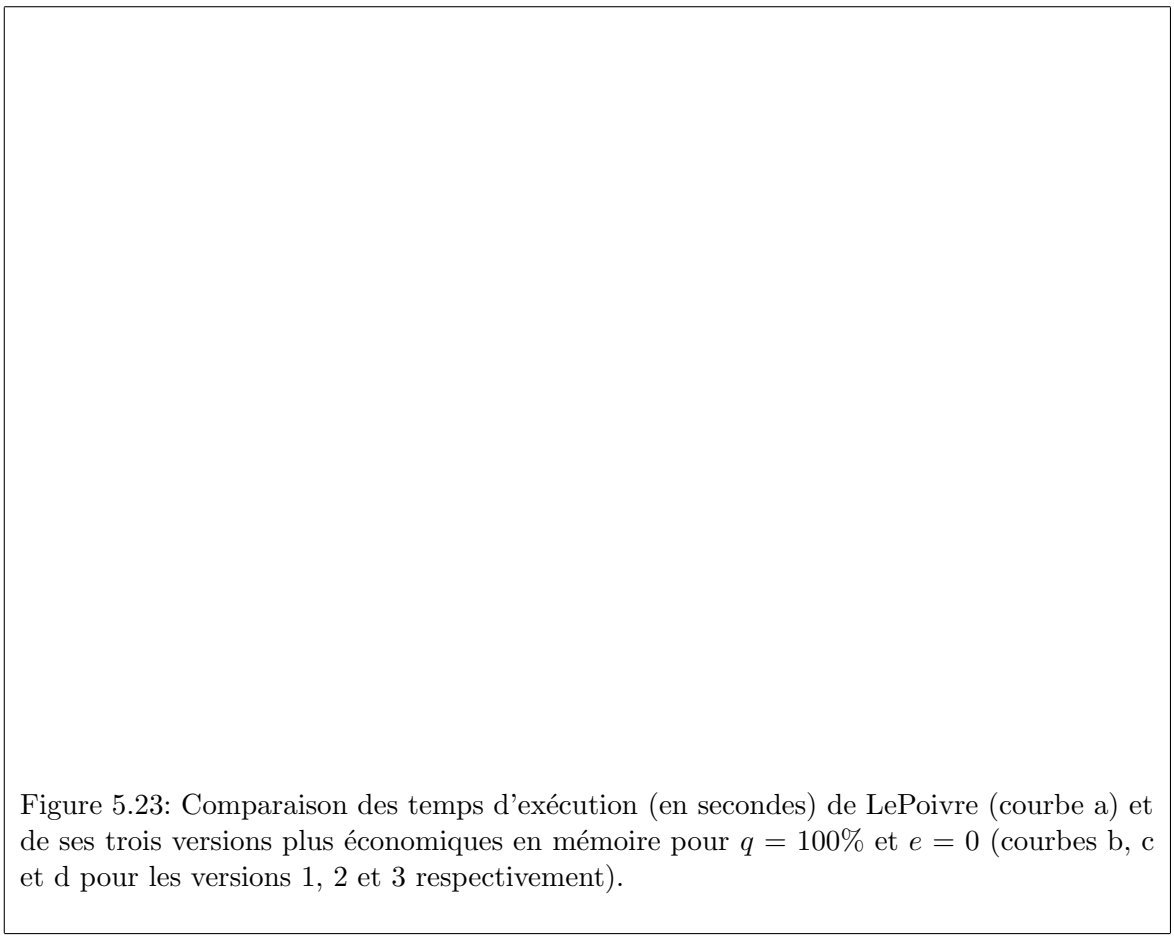


Figure 5.23: Comparaison des temps d'exécution (en secondes) de LePoivre (courbe a) et de ses trois versions plus économiques en mémoire pour $q = 100\%$ et $e = 0$ (courbes b, c et d pour les versions 1, 2 et 3 respectivement).

jeu d'essai a consisté cette fois en 140 chaînes d'ADN de longueur moyenne 100 nucléotides. Ces chaînes correspondent à des promoteurs de *Bacillus subtilis*. Elles sont utilisées à nouveau dans les illustrations et le lecteur est renvoyé au chapitre 6 pour plus de détails les concernant. Il suffit pour l'instant de dire qu'elles présentent toutes 2 à 3 sites impliqués dans le processus de transcription de l'ADN en ARN chez *subtilis*. La couverture utilisée dans le cas de cet exemple a été l'identité. Dans les trois figures, la courbe a fait référence à LePoivre et les courbes b, c et d aux trois versions modifiées de LePoivre.

Ces courbes montrent que le temps d'exécution des versions modifiées 1 et 2 devient d'autant plus grand par rapport à celui de l'algorithme original que le nombre d'erreurs est grand, alors que celui de la version 3 devient d'autant plus petit que ce nombre croît. Le tableau 5.3 donne une indication de cette tendance jusqu'à 7 erreurs pour $q = 100\%$, $N = 10$ et $k = 7$. Observons également qu'à partir de 4 erreurs, le rapport du temps d'exécution de la version modifiée 3 et de LePoivre devient plus grand que la valeur théorique maximale (égale à 2). Une explication à cela est sans doute que les regards qui doivent être en permanence portés sur les ensembles d'occurrences des modèles dans la version 3 sont en pratique moins nombreux que la valeur donnée dans la formule de la complexité (égale à $e + 1$ pour chaque occurrence traitée). En d'autres termes, le calcul de la complexité du pire des cas pour la version modifiée 3 est plus proche de la réalité que celui de l'algorithme LePoivre. Enfin, le tableau 5.4 indique les besoins en mémoire de LePoivre et de cette troisième version modifiée pour $e = 2$, $q = 50\%$, $k = 10$ et

Figure 5.24: Comparaison des temps d'exécution (en secondes) de LePoivre (courbe a) et de ses trois versions plus économiques en mémoire pour $q = 100\%$ et $e = 1$ (courbes b, c et d pour les versions 1, 2 et 3 respectivement).

Figure 5.25: Comparaison des temps d'exécution (en secondes) de LePoivre (courbe a) et de ses trois versions plus économiques en mémoire pour $q = 100\%$ et $e = 2$ (courbes b, c et d pour les versions 1, 2 et 3 respectivement).

Nombre d'erreurs	Temps	
	LePoivre	Modification Version 3
1	1.67	1.20
2	33.06	21.38
3	254.54	145.76
4	1035.63	456.84
5	2631.05	847.09
6	4508.46	1237.58
7	6102.20	1502.38

Tableau 5.3: Comparaison du temps d'exécution (en secondes) de LePoivre et de sa version modifiée 3 pour un nombre d'erreurs croissant, $q = 100\%$, $N = 10$, $k = 7$.

un nombre de chaînes variant entre 20 et 140 avec un pas de 20 (la quantité mesurée correspond au nombre d'occurrences à stocker). L'occupation mémoire des versions 1 et 2 est constante lorsque k est fixé et égale à k et $N.k$ respectivement.

5.4.2.2 Couverture combinatoire pondérée

5.4.2.2.1 Introduction

L'algorithme que nous introduisons maintenant et qui a été présenté dans [Sagot and Viari, 1996] travaille avec une relation de ressemblance différente de celles utilisées par les algorithmes de la section précédente. Elle correspond à la définition 3.5.12 où les modèles représentent des produits d'ensembles d'une couverture combinatoire pondérée. Une telle couverture contient souvent plus d'éléments que les couvertures habituellement employées par les algorithmes précédents et, surtout, son degré de non-transitivité est en moyenne bien plus élevé. Ainsi, dans le cas d'une couverture combinatoire CCP définie sur l'alphabet Σ des nucléotides, CCP contient 15 éléments (tous les sous-ensembles de Σ sauf le sous-ensemble vide) et le degré g de non-transitivité de CCP est égal à 8. Dans ce cas, nous avons également $\overline{g} = 8$.

Les difficultés d'une telle approche doublement combinatoire du problème de la recherche de groupes de mots similaires peuvent donc être considérables. Il est clair en particulier que dans le cas des protéines, où $\mid \Sigma \mid = 20$, la couverture utilisée ne pourra être totalement combinatoire. Nous reviendrons là-dessus plus loin. Pour l'instant cependant, nous allons voir comment traiter une couverture combinatoire complète (tous les sous-ensembles de Σ peuvent être autorisés au moins une fois dans les modèles) dans le cas où Σ est l'alphabet des nucléotides.

L'algorithme est encore une fois présenté de manière progressive. Nous commençons par donner la formule de récurrence permettant de construire les modèles (i.e. leurs ensembles d'occurrences) par longueurs croissantes. Puis nous décrivons un premier algorithme direct pour résoudre le problème général par rapport à la définition de ressemblance adoptée ici. Nous présentons ensuite deux améliorations de cette version initiale qui nous permettent de résoudre le même problème de façon plus efficace et rapide en pratique et/ou en théorie. La première introduit une minimalité gauche-droite des ensembles composant un modèle et la seconde réalise

Nombre de chaînes	Nombre d'occurrences	
	LePoivre	Modification Version 3
20	628798	453830
40	609136	429261
60	639261	448940
80	551173	386660
100	625088	438628
120	731256	510860
140	708499	495232

Tableau 5.4: Mémoire occupée (indiquée ici en nombre d'occurrences devant être stockées) par LePoivre et sa version modifiée 3 pour $e = 2$, $q = 50\%$, $k = 10$ et un nombre de chaînes variant entre 20 et 140.

une esquisse de l'espace solution avant de l'explorer en détail. Dans ce second cas, l'amélioration obtenue dans le temps d'exécution peut parfois être importante.

5.4.2.2.2 Définition de récurrence

Soient $s \in \Sigma^*$, $X \in \Sigma$ et i une position dans s. Soient $CCP = \{(S, w_S)\}$ une couverture combinatoire pondérée de Σ et $M = M'S \in CCP^k$. Nous avons alors le lemme suivant :

Lemme 5.4.5

$$(i, u = u'X) \in Occ(M = M'S)$$

$$\Longleftrightarrow$$

$$(i, u') \in Occ(M'),\ X \in S,\ (p(S, M') + 1) \leq w_S$$

où $p(S, M')$ est le nombre de fois où l'ensemble S apparaît dans le modèle M'.

Ce lemme fournit une façon simple de construire l'ensemble $Occ(M)$ à partir de l'ensemble $Occ(M')$ où $\mid M' \mid = \mid M \mid - 1$. L'ensemble $Occ(M)$ pour $\mid M \mid = 1$ et $w_M \geq 1$ est obtenu en glissant M le long de la chaîne s et en plaçant dans $Occ(M)$ tous les éléments (i, s_i) pour lesquels $s_i \in M$. Bien sûr, si $w_M = 0$, c'est-à-dire, si l'ensemble M n'est pas autorisé dans les modèles, alors $Occ(M) = \emptyset$.

5.4.2.2.3 Algorithme direct

5.4.2.2.3.1 Algorithme

Une idée de l'algorithme direct pour résoudre le Problème 1 est donnée dans la figure 5.26. La résolution du Problème 2 ne nécessite qu'une modification mineure au début de la fonction Explore comme pour les algorithmes précédents.

```
/* Entrée */
        s = s1s2...sN : une chaîne de longueur n = concaténation des chaînes s1, s2, ..., sN
        CCP = {{S}, w_S} : une couverture combinatoire de Σ
        q : contrainte de quorum
        k : une longueur fixe
/* Sortie */
        Tous les modèles M de longueur k ayant des instances dans au moins q des chaînes
/* Principales structures de données */
        M = produit d'ensembles de la couverture CCP
        Occ(M) = ensemble des occurrences d'un modèle M : implémenté comme une pile
        p_M = tableau de taille 2^|Σ| defini par p_M[S] = nombre de fois où l'ensemble
        S apparaît dans l'ensemble M
        Modèles[i] = ensembles de CCP auxquels s_i appartient : implémenté comme une
        pile (cette structure n'est pas vraiment nécessaire)
        ExtensionPossible : comme pour les algorithmes précédents
Main {
        pour (i ∈ [1..n])
                pour (j ∈ [1..2^|Σ|])
                        si ((w_{S_j} ≥ 1) et (i ∈ S_j)) {
                                p_{S_j}[S_j] = 1;
                                Occ(S_j) = Occ(S_j) ∪ {i};
                                Modèles[i] = Modèle[i] ∪ S_j;
                        }
        pour (j ∈ [1..2^|Σ|])
                si (Occ(j) vérifie quorum de q)
                        Explore(S_j, 1, Occ(S_j), p_M);
}
Explore(M, l, Occ(M), p_M) {
        si (l = k)                                      /* fin de la récursion */
                stocke M et Occ(M);
        sinon {                                         /* pas de la récursion */
                pour (i ∈ Occ(M))
                        pour (M_1 ∈ Modèles[i + l])
                                si ((p_M[M_1] + 1) ≤ w_{M_1}) {
                                        p_{MM_1}[M_1] = p_M[M_1] + 1;
                                        Occ(MM_1) = Occ(MM_1) ∪ {i};
                                        ExtensionPossible = ExtensionPossible ∪ {M_1};
                                }
                pour (M_1 ∈ ExtensionPossible)
                        si (Occ(MM_1) vérifie quorum de q)
                                Explore(MM_1, l + 1, Occ(MM_1), p_{MM_1});
        }
}
```

Figure 5.26: Algorithme direct (couverture combinatoire pondérée) pour résoudre le Problème 1.

5.4.2.2.3.2 Complexité

La complexité qui est obtenue ici est la même que celle observée pour l'algorithme Poivre (section 5.4.2.1.2), simplement la couverture est définie différemment. Cette complexité est donc $O(n.k)$ en espace et $O(n.N.k.g^k)$ en temps, avec cependant cette fois, g (ainsi que $\overline{g}$) égal à 8. Rappelons que, dans le cas de LePoivre, nous travaillions avec des valeurs de $\overline{g}$ proches de l'unité ($\overline{g} \sim 1.2$)

Bien sûr, une fois encore le quorum q n'est pas pris en considération (c'est-à-dire, nous faisons comme si $q = 1$). De plus, nous considérons également dans ce calcul que $w_S = \infty$ pour tout $S \in \mathcal{P}^+(\Sigma)$.

Malgré cela, une complexité proportionnelle à g^k lorsque $g = 8$ est excessive, et nous présentons donc dans la prochaine section une première amélioration de cet algorithme direct.

5.4.2.2.4 Minimalité gauche-droite

5.4.2.2.4.1 Idée principale

Considérons l'exemple suivant :

Exemple 5.4.1 *Soient q = 3,*
$s1 = ACAA$
$s2 = CAGA$
$s3 = AGAA$
$s4 = CTGA$
et :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 1) \end{array} \right\}.$$

Alors $M_1 = \{A, C\}\{A, C, G, T\}\{A, G\}\{A, T\}$ et $M_2 = \{A, C\}\{A, C, G, T\}\{A, G\}\{A\}$ sont tous deux des modèles valides de longueur 4. Cependant le modèle M_1 n'apporte aucune information supplémentaire par rapport au modèle M_2 puisque $Occ(M_1) = Occ(M_2)$ et $M_2 \subseteq M_1$ (où $(M_2 = \prod_{i=1}^{k} S_2^i) \subseteq (M_1 = \prod_{i=1}^{k} S_1^i)$ si $S_2^i \subseteq S_1^i$ pour $1 \leq i \leq k$).

Ceci demeure vrai pour tous les modèles M_1' ayant le modèle M_1 comme préfixe propre : l'information qu'ils apportent est superflue par rapport à celle contenue dans les modèles M_2' de même longueur ayant M_2 comme préfixe propre. Aucune information n'est donc perdue si le sous-arbre de l'arbre des modèles ayant M_1 pour racine n'est jamais traversé.

Observons que l'exemple précédent est très différent de celui indiqué ci-dessous :

Exemple 5.4.2 *Soient CCP et q comme avant et maintenant :*
$s1 = ACAA$
$s2 = CAGA$
$s3 = AGAA$
$s4 = ATGA$.
Dans ce cas, $M_1 = \{A, C\}\{A, C, G, T\}\{A, G\}\{A\}$, $M_2 = \{A\}\{A, C, G, T\}\{A, G\}\{A\}$ et $M_3 = \{A\}\{A, C, G, T\}\{A, G\}\{A, T\}$ sont tous des modèles valides de longueur 4 mais si M_3 est superflu par rapport à M_2 puisque $Occ(M_3) = Occ(M_2)$ et $M_2 \subseteq M_3$, cela n'est plus vrai pour le

modèle M_1 par rapport à M_2. La raison en est que, si comme produit d'ensembles, nous avons bien $M_2 \subseteq M_1$, nous avons aussi par ailleurs que $Occ(M_1) \neq Occ(M_2)$.

Les modèles M_1 et M_2 peuvent donc conduire à des solutions finales différentes qui sont toutes deux valides et l'arbre des modèles ne peut être élagué au nœud étiqueté par aucun de ces modèles.

Ces exemples montrent que, étant donné deux modèles valides M_1 et M_2, si les deux conditions suivantes sont vérifiées simultanément :

c1 $Occ(M_1) = Occ(M_2)$

c2 $M_2 \subseteq M_1$

alors l'information apportée par le modèle M_1 est superflue en termes aussi bien d'un produit d'ensembles de la couverture que de ses occurrences. Il peut donc être éliminé, et l'arbre des modèles élagué au nœud étiqueté par ce modèle. En d'autres termes, lorsqu'à une étape donnée de l'algorithme, les ensembles d'occurrences de deux modèles sont identiques, nous n'avons besoin de conserver que celui qui correspond à un produit d'ensembles minimaux par rapport à l'autre. Cela peut représenter une économie supplémentaire en temps (et en espace).

En pratique cependant, les deux conditions ci-dessus ne sont pas celles que nous vérifions effectivement. Au lieu de cela, ce qui est vérifié est si, étant donné deux modèles $M_1 = MS_1$ et $M_2 = MS_2$, la condition suivante est vrai :

c3 $Occ(M) \cap Occ(S_1) = Occ(M) \cap Occ(S_2)$ (il s'agit de la même condition que c1 écrite d'une autre façon)

c4 $S_2 \subseteq S_1$

Ceci signifie que la minimalité des ensembles dans les modèles est toujours relative aux derniers ensembles $S \in \mathcal{P}^+(\Sigma)$ qui sont successivement concaténés à un même modèle M, cette minimalité n'est jamais vérifiée par rapport à tous les modèles. Clairement, c4 est une condition plus faible que c2. En voici un exemple :

Exemple 5.4.3 *Soient CCP comme dans l'exemple 5.4.1, $q = 3$ et :*

$$s1 = ACAA$$
$$s2 = CAGT$$
$$s3 = AGAA$$
$$s4 = ATGA$$

Alors $M_1 = \{A,C\}\{A,C,G,T\}\{A,G\}$ et $M_2 = \{A\}\{A,C,G,T\}\{A,G\}$ sont tous deux des modèles valides et non superflus de longueur 3 et les deux doivent être préservés. Les modèles $M_3 = \{A\}\{A,C,G,T\}\{A,G\}\{A\}$ et $M_4 = \{A,C\}\{A,C,G,T\}\{A,G\}\{A\}$ de longueur 4 sont aussi valides mais M_4 est superflu car $Occ(M_3) = Occ(M_4)$ et $M_3 \subseteq M_4$. Dans notre implémentation cependant, tous deux sont conservés puisque $M_3 = M_2\{A\}$, $M_4 = M_1\{A\}$ et $M_2 \neq M_1$. Aussi la minimalité de M_3 par rapport à M_4 ne peut être vérifiée.

La vérification des conditions c3 et c4 plutôt que celle de c1 et c2 est nécessaire car les modèles sont construits en profondeur. Le désir d'éviter un parcours en largeur de l'arbre n'est cependant pas la seule raison pour laquelle nous ne cherchons pas pour l'instant à vérifier les conditions c1 et c2 totalement. Une seconde motivation est que cette vérification serait impossible à réaliser sans que cela ajoute à la complexité de l'algorithme car elle impliquerait des comparaisons entre tous les modèles d'une certaine longueur. Enfin une troisième et dernière

motivation également importante est qu'en fait même un parcours en largeur ne permettrait pas de vérifier c1 et c2. En effet, certains modèles minimaux ne vont jamais être engendrés. Illustrons cela sur un exemple :

Exemple 5.4.4 *Soient CCP comme dans l'exemple 5.4.1,* $q = 4$ *et :*

$s1 = CAAA$
$s2 = ATAA$
$s3 = AAGA$
$s4 = AAAC$

Alors $M_1 = \{A, C\}\{A, T\}\{A, G\}\{A, C, G, T\}$ *et* $M_2 = \{A, C, G, T\}\{A, T\}\{A, G\}\{A, C\}$ *sont tous deux des modèles valides et non superflus de longueur* 4 *puisque, si* $Occ(M_1) = Occ(M_2)$*, aucune des deux inclusions* $M_1 \subseteq M_2$ *ou* $M_2 \subseteq M_1$ *n'est vérifiée. Tous deux sont donc, dans ce sens, minimaux. Ces modèles devraient alors être conservés, s'ils étaient engendrés. Mais quel que soit le type de parcours effectué dans l'arbre des modèles, le modèle* M_2 *ne sera jamais créé (en tant qu'étiquette seulement puisqu'en tant qu'ensemble d'occurrences il apparaît sous le 'nom' de* $\{A, C\}\{A, T\}\{A, G\}\{A, C, G, T\}$*) parce que la construction des modèles se fait, toujours, de la gauche vers la droite!*

Il est important de comprendre que ces modèles qui ne sont pas engendrés en tant qu'étiquettes n'invalident pas l'affirmation que l'algorithme est capable de trouver tous les ensembles d'occurrences correspondant à des modèles valides — la recherche des groupes de mots similaires suivant la définition 3.5.12 de ressemblance demeure donc exhaustive par rapport à ces ensembles et non par rapport aux modèles. En d'autres termes, s'il arrive qu'un modèle soit 'oublié', alors il existe nécessairement un autre modèle qui présente le même ensemble d'occurrences.

Il y a ainsi deux facteurs qui font que la minimalité des modèles que l'algorithme construit, à travers la vérification des conditions c3 et c4, n'est pas absolue. Le premier est lié au fait que nous parcourons l'arbre en profondeur. Des modèles non minimaux (superflus) peuvent être conservés. Le second facteur découle du fait que le processus de construction possède une direction dans le temps. Celle-ci est elle-même conséquence de la pondération finie attribuée à certains éléments de la couverture. Elle est donc intrinsèque au principe de développement par récurrence de l'algorithme.

La propriété qui est vérifiée dans tous les cas par les conditions c3 et c4 est alors ce que nous avons appelé une minimalité gauche-droite. Nous pouvons affirmer aussi que si un modèle M donné est conservé, tous les modèles de plus courtes longueurs en ordre lexicographique de cardinalité des ensembles les composant qui vérifient le quorum sont également préservés. Ceci est le cas de M_4 et M_3 dans l'exemple 5.4.3.

La question maintenant est, comment vérifier ces conditions de manière efficace?

Soit M' le modèle que nous sommes en train d'allonger d'une unité vers la droite. Soient $Occ(M')$ l'ensemble de ses occurrences et $S = \{s_{i+|M'|} \in \Sigma \mid i \in Occ(M')\}$. En d'autres termes, S est l'ensemble des symboles de Σ qui suivent chacune des occurrences de M' dans s.

Alors les modèles $M = M'S'$ pour $S' \in CCP$ qui doivent être préservés sont ceux pour lesquels :

1. $(p(S', M') + 1) \leq w_{S'}$
   et
2. (a) $(S' \subseteq S)$
   ou

Soit M' le modèle que nous essayons d'étendre d'une unité vers la droite;
Soit $S = \{s_{i+|M'|} \in \Sigma \mid i \in Occ(M')\}$;
Soit Sup(S') l'union de tous les ensembles S'' tels que $S' \subset S''$ et $\mid S'' \mid = \mid S' \mid + 1$;
EnsembleValable = $\emptyset$;

pour chaque ensemble $S' \in CCP$ pris par cardinalité croissante
    si $(S' \subseteq S)$
        si $((p(S', M') + 1) \leq w_{S'})$
            $M = M'S'$ est un modèle valide;
        sinon
            EnsembleValable = EnsembleValable $\cup$ Sup(S');
    sinon si $(S' \in$ EnsembleValable$)$
        si $((p(S', M') + 1) \leq w_{S'})$
            $M = M'S'$ est un modèle valide;
        sinon
            EnsembleValable = EnsembleValable $\cup$ Sup(S');

Figure 5.27: Vérification des conditions c3 et c4 (couverture combinatoire pondérée — première amélioration de l'algorithme direct).

(b) S' est un plus petit ensemble tel que $S \subset S'$ et tel qu'il existe $S'' \subseteq S$ avec $(p(S'', M') + 1) > w_{S''}$.

Bien sûr, $M'S'$ doit en outre vérifier la contrainte de quorum. Observons aussi qu'il peut y avoir plus d'un ensemble S' verifiant 2(b).

5.4.2.2.4.2 Algorithme

Une idée de l'algorithme correspondant à l'implémentation des deux points c3 et c4 ci-dessus est donné dans la figure 5.27.

5.4.2.2.4.3 Complexité

Si les structures de données Sup(S) pour tout S dans la couverture et EnsembleValable sont implémentées comme des tableaux de booléens et si les opérations d'union et d'appartenance sont également booléennes, alors la complexité en temps que demande la vérification des conditions c3 et c4 est $O(p)$. Elle est donc linéaire avec le nombre des éléments de la couverture. La complexité en temps totale de l'algorithme est ainsi majorée par $O(n.N.k.g^k.p)$. Celle en espace demeure la même que pour l'approche directe.

Bien que la complexité en temps soit plus mauvaise en théorie, en pratique cela n'est pas le cas, la raison étant que moins de modèles sont produits et que l'espace de recherche est plus réduit. Des tests ont montré (voir la section performance) que le facteur en temps gagné dans l'exécution de l'algorithme est en général égal à 4 pour les acides nucléiques (cette version s'exécute 4 fois plus rapidement). Nous introduisons dans la section suivante une seconde idée qui peut améliorer encore plus la performance de l'algorithme, parfois de manière importante.

5.4.2.2.5 Esquisse de l'espace solution

5.4.2.2.5.1 Idée principale

La seconde idée pour améliorer la performance de l'algorithme (à la fois théorique et pratique) est issue de l'expérience suivante. Nous avions lancé l'algorithme à la recherche des plus longs modèles présents à 100% dans 4 chaînes aléatoires de longueur 100 définies sur l'alphabet $\{A, C, G, T\}$ avec la couverture :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (S, 1)\ \forall S \in \mathcal{P}^+(\Sigma),\ \mid S \mid \neq 1 \end{array} \right\}.$$

L'algorithme a nécessité 875 secondes sur une station de travail Silicon Graphics (R4000) pour produire une réponse, même après que la première modification (section 5.4.2.2.4) ait été implémentée.

Nous avions alors relancé l'algorithme dans les mêmes conditions que précédemment, avec cette fois :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (\{A, C, G, T\}, 2^{|\Sigma|-1} - \mid \Sigma \mid) \\ (S, 0)\ \forall S \in \mathcal{P}^+(\Sigma),\ \mid S \mid \neq 1\ et\ S \neq \Sigma \end{array} \right\}$$

c'est-à-dire, les seuls ensembles permis dans les modèles étaient les ensembles unitaires et le joker (en anglais, 'wild card' ou 'don't care symbol'), ce dernier ensemble étant autorisé au plus 11 (= 15 - 4) fois dans les modèles. De plus, nous avions recherché tous les modèles de longueur $k = 15$, qui correspondait à la longueur des plus longs modèles trouvés dans le premier cas.

En termes d'ensembles d'occurrences, il est clair que nous avons : espace solution(expérience 1) $\subseteq$ espace solution(expérience 2).

Le temps mis par l'algorithme pour produire une réponse à l'expérience 2 avait été de 2 secondes sur la même machine.

En termes d'une recherche dans l'espace de tous les modèles, l'expérience 2 peut être vue comme effectuant une exploration à 'gros grain' de cette espace, dans laquelle plusieurs branches de l'arbre sont examinées ensembles (en une seule fois). Cette expérience simple montre qu'une telle approche peut parfois être plus efficace que celle qui consiste à essayer de rechercher les réponses 'fines' dès le départ. Si cette première exploration conserve strictement le contour général de l'espace solution tout en étant capable d'éliminer une grande partie de l'espace de recherche initial, alors dans une seconde phase de l'algorithme, nous n'avons plus besoin d'explorer que le sous-espace délimité par la première phase. Cette manière de faire peut être avantageuse si l'exploration à gros grain est réalisable de façon plus rapide que l'exploration à grain fin et si l'espace solution qui est esquissé est plus réduit que l'espace de départ, ou si la première exploration plus 'grossière' est capable non seulement de limiter cet espace mais également de l'organiser d'une certaine manière.

5.4.2.2.5.2 Algorithme

Étant donné la couverture CCP définie par :

$$CCP = \left\{ \begin{array}{l} (\{X\}, a)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, b)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, c)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, d) \end{array} \right\}$$

1. Résolution du Problème 1 — recherche de tous les modèles de longueur k satisfaisant le quorum q :

1.1. réaliser une première recherche avec la couverture CCP' définie par :

$$CCP' = \left\{ \begin{array}{l} (\{X\}, a)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 0)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, b + c + d) \end{array} \right\}$$

1.2. pour chaque modèle M trouvé lors de la première recherche
concaténer ses occurrences et lancer une version légèrement modifiée de l'algorithme (voir la section 5.4.2.2.5.2), cette fois avec la couverture CCP et en recherchant tous les modèles valides de longueur k

2. Résolution du Problème 2 — recherche de tous les modèles de plus grande longueur satisfaisant le quorum q :

2.1. réaliser une première recherche avec la couverture CCP' donnée ci-dessus
soit k'_{max} la longueur des plus longs modèles trouvés

2.2. pour chaque modèle M trouvé lors de la première recherche
concaténer ses occurrences et lancer une version légèrement modifiée de l'algorithme (voir la section 5.4.2.2.5.2) avec cette fois la couverture CCP et en recherchant tous les modèles valides de longueur k'_{max}

2.3. si aucun modèle valide n'est trouvé
faire {
résoudre le Problème 1 pour $k = k'_{max}$ - 1, puis $k = k'_{max}$ - 2 etc
}
jusqu'à ce qu'au moins un modèle valide soit trouvé
(observons que cette dernière étape peut aussi être réalisée de manière dichotomique)

Figure 5.28: Une idée de la version en deux phases de recherche de l'algorithme Combi (couverture combinatoire pondérée — seconde amélioration de l'algorithme direct).

où a, b, c, d sont des entiers non-négatifs, et dans le cas de b, c, et d, finis, le schéma à suivre afin de résoudre le Problème 1 ou le 2 est celui indiqué dans la figure 5.28. L'algorithme final est appelé Combi.

La modification à apporter à l'algorithme dans la seconde phase (points 1.2 et 2.2) concerne uniquement la première étape, c'est-à-dire la construction des ensembles $Occ(M)$ pour $\mid M \mid = 1$ et $w_M \geq 1$. Ces ensembles sont maintenant obtenus en glissant le long de la chaîne s, mais cette fois seules sont placées dans $Occ(M)$ les positions i pour lesquelles $s_i \in M$ et i est un multiple de k (puisque nous travaillons dans cette seconde étape avec une chaîne formée par la concaténation des occurrences).

En reprennant nos expériences précédentes, si nous exécutons cette version en deux phases de l'algorithme sur le même ensemble de 4 chaînes aléatoires de longueur 100 chacune et recherchons les plus longs modèles présents dans 100% des chaînes, nous obtenons la réponse correcte en 38 secondes, ce qui représente un gain d'un facteur 25 par rapport à la version antérieure.

La version finale est donc 100 fois plus rapide que l'approche directe sur cet exemple particulier (rappelons que la première amélioration allait déjà 4 fois plus vite). Des tests supplémentaires (voir section performance) indiquent le gain observé sur d'autres exemples. Ils montrent que celui-ci dépend beaucoup des données, alors que le gain obtenu par l'exigence d'une minimalité gauche-droite est plus constant.

5.4.2.2.5.3 Complexité

Il est aisé de voir que la complexité en temps de la première phase de l'algorithme est $O(n.N.k.2^k)$ (g ainsi que $\overline{g}$ sont ici tous deux égaux à 2) et que celui de la seconde phase est $O(m.k.p)$ où m est le nombre total d'occurrences trouvées dans la première phase. En général, m est bien plus petit que $n.N.g^k$ (voir la section performance). Comme les deux phases de l'algorithme doivent être répétées au plus k'_{max} fois où k'_{max} est la plus grande longueur des modèles trouvés avec la couverture CCP', la complexité totale en temps de l'algorithme est bornée supérieurement par $O(n.N.(k'_{max})^2.2^{k'_{max}} + m.(k'_{max})^2.p)$.

En réalité, la valeur de k_{max} par rapport à la couverture CCP peut être déterminée de manière dichotomique à partir de la valeur k'_{max} par rapport à la couverture CCP'. La borne supérieure pour la complexité est donc $O(n.N.k'_{max}.\log k'_{max}.2^{k'_{max}} + m.k'_{max}.\log k'_{max}.p)$. Une telle technique n'est en fait pas utile en général (voir performance) car k_{max} est le plus souvent proche de k'_{max}. Lorsque des modèles de longueur fixe k sont recherchés, cette borne devient $O(n.N.k.2^k + m.k.p)$.

Comme pour l'approche directe, l'introduction d'une contrainte de quorum $q > 1$ et le fait que $w_S \neq \infty$ pour la plupart des ensembles S de la couverture, signifient qu'en pratique le comportement moyen de l'algorithme est bien meilleur que cela (voir ci-dessous).

Observons enfin que la complexité en espace de cette version finale de l'algorithme est bornée supérieurement par $O(n.N.k'_{max}.2^{k'_{max}})$ puisque tous les modèles de longueur k'_{max} de la première phase ainsi que leurs occurrences doivent être conservés pour la seconde phase.

5.4.2.2.5.4 Observation importante concernant les modèles

Il est important de noter que, jusqu'à maintenant, nous n'avons pas tenu compte de quels sous-ensembles pouvaient être présents en première et dernière positions d'un modèle considéré valide. En particulier, un tel modèle peut très bien être de la forme $M = \{\Sigma\}M'\{\Sigma\}$, c'est-à-dire débuter et se terminer par le joker. Il est clair que de tels modèles ne sont souvent pas très intéressants, et il est relativement facile de modifier l'algorithme de telle sorte que les modèles repérés (d'une longueur fixe ou de la plus grande longueur possible) ne contiennent le joker ni en première ni en dernière position.

Exiger l'absence du joker en première position implique modifier l'appel de la fonction Explore à la fin de la fonction principale de l'algorithme 5.26. Cet appel ne se fait désormais que pour les $S_i \in CCP$ tels que $S_i \neq \{\Sigma\}$. Dans le cas où des modèles d'une longueur fixe sont recherchés, ne sont stockés que les modèles $M = M'S$ pour lesquels $S \neq \{\Sigma\}$. Enfin, lorsque ce sont les plus longs modèles qui nous intéressent, les modèles de longueur plus petite conservés au préalable ne sont rejetés que si le modèle valide nouveau ne possède pas de joker en dernière position.

Cette contrainte imposée au modèle ne change rien à la complexité théorique de l'algorithme mais elle peut avoir une influence relativement importante sur le temps d'exécution en pratique ainsi que sur le volume du résultat final obtenu. Cela est montré dans la section performance.

5.4.2.2.6 Variantes : introduction de contraintes supplémentaires

5.4.2.2.6.1 Pondération sur les ensembles limitée par la pondération sur le joker

Travailler avec un algorithme en deux phases nous fournit une manière facile d'imposer des contraintes supplémentaires sur le nombre d'ensembles qui peuvent apparaître dans les modèles valides outre celle concernant l'absence de joker en première et dernière position (voir la section 5.4.2.2.5.4). Ceci est obtenu en fixant le poids attaché à $\{\Sigma\}$ dans CCP' non pas égal à $b+c+d$ mais égal plutôt à une valeur notée f strictement plus petite que $b+c+d$. Dans ce cas, les modèles ne peuvent contenir plus de f ensembles 'spéciaux', où par un ensemble 'spécial' nous entendons n'importe quel ensemble non unitaire spécifié dans la seconde phase. Cette contrainte additionnelle a prouvé son utilité dans un certain nombre d'applications (voir le chapitre 6). Un cas particulier très employé en pratique consiste à établir $b=c=d=f$. Ceci équivaut à permettre la présence de f sous-ensembles non unitaires dans les modèles, quels qu'ils soient.

5.4.2.2.6.2 Pondération sur les ensembles groupés par cardinalité

Dans ce cas, les pondérations sont affectées aux ensembles de la couverture groupés par cardinalité. Un exemple d'une telle pondération est donnée ci-dessous.

Exemple 5.4.5 *Soient* $\Sigma = \{A, C, G, T\}$ *et :*

$$CCP_{card} = \left\{ \begin{array}{l} (S, \infty) \; si \;\; \mid S \mid = 1, \\ (S, 3) \; si \;\; \mid S \mid = 2, \\ (S, 2) \; si \;\; \mid S \mid = 3, \\ (\{A, C, G, T\}, 1) \end{array} \right\}.$$

En d'autres termes, les sous-ensembles de $\{\Sigma\}$ ayant 2 symboles sont permis 3 fois dans les modèles, ceux ayant 3 symboles le sont 2 fois et le joker 1 fois. Le modèle $\{A,C\}\{A,C\}\{A,C\}$ $\{A,C,G,T\}\{A,C,G\}\{A,C,G\}$ est ainsi un modèle possible, c'est-à-dire est un élément de CCP_{card}. La modification à apporter à l'algorithme 5.26 est immédiate à réaliser : le tableau p_M compte désormais la présence des sous-ensembles par leur cardinalité et doit donc avoir une taille égale à $\mid \Sigma \mid$.

5.4.2.2.6.3 Nombre de jokers variable décrémenté à chaque retour à la première phase

Enfin, dans une dernière variante de l'algorithme, le nombre de jokers autorisés lors de la première phase peut être décrémenté à chaque étape. Ainsi initialement il est fixé égal à $b+c+d$, cependant si la seconde phase ne réussit à identifier aucun modèle valide et que l'algorithme doit revenir à la première, ce nombre est fixé à $(b+c+d)-1$ et ainsi de suite. Le nombre de jokers varie donc avec k'_{max}. Bien évidemment , cette variante est une heuristique. Elle trouve sa justification dans le fait que si aucun modèle de longueur k_{max} n'a pu être trouvé dans la seconde phase à partir de ceux repérés dans la première, alors il y a relativement peu de chances pour qu'un modèle de longueur $k'_{max}-1$ de la première phase et ayant le même nombre $b+c+d$ de jokers qu'au départ résulte en un modèle valide final. Nous verrons dans la section performance que cette heuristique fonctionne correctement la plupart du temps (i.e. elle permet de trouver les mêmes modèles que la variante exacte). Elle a tendance à être prise en défaut pour des faibles quorums.

5.4.2.2.6.4 Pondération sur les ensembles nulle presque partout (cas des protéines)

Ainsi que nous l'avons mentionné en introduction à cette section, il n'est pas envisageable, du moins pour l'instant, de travailler avec une couverture totalement combinatoire lorsque nous voulons comparer des protéines. Dans ce cas en effet, $\mid \Sigma \mid = 20$ et, par conséquent, $\mid \mathcal{P}^+(\Sigma) \mid = (2^{20} - 1)$.

Plusieurs approches peuvent être considérées afin de réussir néanmoins à traiter le cas de ces macromolécules. Toutes reposent sur le même principe, à savoir que la pondération affectée aux sous-ensembles de l'alphabet des acides aminés sera nulle presque partout sauf pour un sous-ensemble limité d'éléments de $\mathcal{P}^+(\Sigma)$. C'est dans le choix de ce sous-ensemble que les méthodes varient.

Celle adoptée par Neuwald [Neuwald and Green, 1994] consiste à établir que la pondération d'un sous-ensemble S de $\mathcal{P}^+(\Sigma)$ ne sera strictement positive que si :

- $S = \{\Sigma\}$;
  ou
- $S = \{a\}$ avec $a \in \Sigma$;
  ou encore
- $S = \{a, b\}$ avec $a, b \in \Sigma$ et $\mathcal{M}(a, b) \geq t$ où $\mathcal{M}$ est une matrice de similarité (par exemple PAM) et t est une valeur seuil fixée à l'avance.

Neuwald n'utilise en fait pas le concept d'une couverture combinatoire pondérée mais une contrainte très similaire est employée. L'idée d'un modèle composé de produits de certains sous-ensembles de l'alphabet des acides aminés apparaît également clairement sous le nom de motif. Ce dernier doit vérifier les conditions suivantes :

1. un motif valide doit débuter et se terminer par un sous-ensemble unitaire;
2. un motif valide doit comporter un nombre fixe de sous-ensembles de cardinalité 1 ou 2;
3. la longueur maximale des motifs est limitée ce qui, en même temps que la condition 2, limite effectivement le nombre de jokers permis.

La méthode de construction des motifs est techniquement assez proche de celle que nous suivons (version minimalité), mais la condition d'élagage de l'arbre des motifs est différente. Celle-ci porte en effet non seulement sur une contrainte de quorum q mais aussi sur la signification statistique d'un motif. Ainsi dans leur cas, des critères statistiques influencent la recherche des motifs dès le départ, ce que nous avons voulu éviter de faire pour des raisons qui seront discutées plus loin. Il est très aisé d'appliquer le concept d'une pondération à la couverture avec laquelle Neuwald *et al.* travaillent, ou n'importe quel autre couverture. Pour éviter d'introduire un point de vue comme cela est fait dans le cas de Neuwald à travers l'usage d'une matrice de substitution, il est possible de fixer des critères plus combinatoires afin de décider de la pondération à affecter à un élément de $\mathcal{P}^+(\Sigma)$. Par exemple, nous pouvons établir que ne seront permis dans un modèle que le joker (un nombre fini de fois) ainsi que tous les ensembles de cardinalité inférieure à une certaine valeur c. Il est clair qu'en pratique, c ne peut être très grand.

Nous présentons plus loin une autre façon encore d'aborder la question, qui diffère de celle employée par Neuwald ou suggérée plus haut dans le choix du sous-ensemble et surtout dans le fait que nous allons, contrairement à Neuwald, également pouvoir travailler avec des erreurs.

5.4.2.2.7 Performance en pratique

Nous présentons maintenant les performances de Combi et de ses variantes sur un jeu d'essai composé des mêmes 140 chaînes de promoteurs de *Bacillus subtilis* utilisé dans la section 5.4.2.1.7.

Ces performances sont indiquées sous forme de tableaux et non de courbes cette fois, parce qu'il est important dans ce cas d'indiquer et de discuter le rôle qu'exercent plusieurs autres facteurs sur le comportement de l'algorithme. Ces facteurs sont essentiellement :

- le nombre de fois où l'algorithme doit exécuter les deux phases avant de réussir à obtenir des modèles valides lorsque la longueur des modèles recherchés est la longueur maximale k_{max};
- la longueur maximale k'_{max} des modèles obtenus lors du premier passage de l'algorithme par la phase 1 et celle k_{max} obtenue à la fin;
- le nombre de modèles et d'occurrences observés dans la première et dernière phases de l'algorithme pour $k = k_{max}$.

Les valeurs de ces divers facteurs et les temps d'exécution correspondants (observés sur une Sparc Station 20) sont donnés pour plusieurs couvertures et variantes de Combi. Les couvertures utilisées ont été :

- pour le tableau 5.5 :

$$CCP1 = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 1)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 0) \end{array} \right\}$$

  avec, en première phase :

$$CCP1' = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 0)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 4) \end{array} \right\};$$

- pour le tableau 5.6 :

$$CCP2 = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 1)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 1) \end{array} \right\}$$

  avec, en première phase :

$$CCP2' = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 0)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 4) \end{array} \right\};$$

- pour le tableau 5.7 :

$$CCP3 = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{A, G\}, 2), (\{C, T\}, 2), (\{X, Y\}, 0)\ \textit{pour toute autre paire}\ \{X, Y\} \in \Sigma^2, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 0) \end{array} \right\}$$

avec, en première phase :

$$CCP3' = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 0)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 4) \end{array} \right\}$$

(dans ces trois cas, il s'agit donc de la variante de Combi pour laquelle la pondération sur les ensembles est limitée par la pondération sur le joker en première phase décrite dans la section 5.4.2.2.6.1);

- pour le tableau 5.8 :

$$CCP1_{card} = \left\{ \begin{array}{l} (S, \infty)\ si\ \mid S \mid = 1, \\ (S, 2)\ si\ \mid S \mid = 2, \\ (S, 1)\ si\ \mid S \mid = 3, \\ (\{A, C, G, T\}, 1) \end{array} \right\}$$

avec en première phase :

$$CCP1'_{card} = \left\{ \begin{array}{l} (S, \infty)\ si\ \mid S \mid = 1, \\ (S, 0)\ si\ \mid S \mid = 2, \\ (S, 0)\ si\ \mid S \mid = 3, \\ (\{A, C, G, T\}, 4) \end{array} \right\};$$

- pour le tableau 5.9 :

$$CCP2_{card} = \left\{ \begin{array}{l} (S, \infty)\ si\ \mid S \mid = 1, \\ (S, 2)\ si\ \mid S \mid = 2, \\ (S, 0)\ si\ \mid S \mid = 3, \\ (\{A, C, G, T\}, 1) \end{array} \right\}$$

avec en première phase :

$$CCP2'_{card} = \left\{ \begin{array}{l} (S, \infty)\ si\ \mid S \mid = 1, \\ (S, 0)\ si\ \mid S \mid = 2, \\ (S, 0)\ si\ \mid S \mid = 3, \\ (\{A, C, G, T\}, 2) \end{array} \right\};$$

- pour le tableau 5.10 :

$$CCP3_{card} = \left\{ \begin{array}{l} (S, \infty)\ si\ \mid S \mid = 1, \\ (S, 2)\ si\ \mid S \mid = 2, \\ (S, 2)\ si\ \mid S \mid = 3, \\ (\{A, C, G, T\}, 0) \end{array} \right\}$$

avec en première phase :

$$CCP3'_{card} = \left\{ \begin{array}{l} (S, \infty)\ si\ \mid S \mid = 1, \\ (S, 0)\ si\ \mid S \mid = 2, \\ (S, 0)\ si\ \mid S \mid = 3, \\ (\{A, C, G, T\}, 4) \end{array} \right\}$$

(ces trois séries de tableaux analysent donc le comportement de la variante de Combi décrite dans la section 5.4.2.2.6.2 pour laquelle la pondération porte sur les ensembles groupés par cardinalité; le tableau 5.9 combine cette variante à celle de la section 5.4.2.2.6.1);

- enfin, pour les tableaux 5.11 et 5.12 :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 1)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 1) \end{array} \right\}$$

avec, en première phase :

$$CCP' = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 0)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 11) \end{array} \right\}$$

Ces tableaux appelent plusieurs commentaires. En ce qui concerne les tableaux 5.5 à 5.10, nous constatons que le nombre de modèles obtenus peut varier considérablement avec le nombre de chaînes pour une même valeur de quorum, et cela d'une manière qui est, malheureusement, tout à fait imprévisible. La même variation serait observée si l'on maintenait le nombre de chaînes fixe et que l'on faisait varier cette fois le quorum. Décrémenter l'un ou l'autre augmente en général le nombre de modèles jusqu'au moment où des modèles plus longs (et moins nombreux) sont identifiés. Le degré de non-transitivité de la couverture possède également une grande influence sur le nombre des modèles. En effet, lorsque ce degré est élevé, on observe beaucoup plus de modèles qui sont des variantes proches les unes des autres. La raison en est que plusieurs éléments de la couverture peuvent être utilisés aux positions les plus variables des ensembles d'occurrences. De plus, lorsque le quorum n'est pas égal à 100% (comme c'est le cas ici), chacun de ces éléments peut conduire à une version légèrement modifiée d'un modèle. En ce qui concerne les applications que nous visons, ce type de comportement peut ne pas être négatif. Un exemple de ceci sera donné dans le chapitre des illustrations. Cet exemple suggère cependant qu'il serait intéressant dans certains cas d'ajouter une étape à l'algorithme, dans laquelle les modèles obtenus par l'algorithme tel qu'il est conçu aujourd'hui pourraient être regroupés en des objets plus généraux (des méta-modèles), c'est-à-dire, des modèles de modèles. Une autre possibilité serait de réaliser une évaluation statistique de ces modèles et de les classer ensuite selon leur 'pertinence'.

Toujours en ce qui concerne ces six tableaux, observons que, ainsi que nous l'avions mentionné à la fin de la section 5.4.2.2.6.3, la contrainte qui consiste à décrémenter le nombre de jokers à chaque retour à la première phase de l'algorithme peut avoir une influence importante sur le temps d'exécution. Nous pouvons également vérifier dans ces tableaux que le nombre total d'occurrences trouvées dans la première phase est, d'une part bien inférieur à $n.N.g^k$ et, d'autre part, généralement inférieur au nombre total d'occurrences trouvées en première phase.

Couverture $CCP1$ - pas de décrémentation du nombre de jokers entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	2	10	9	225	284	2182	2141	5.19
50	2	9	8	516	36	44121	2003	26.95
100	3	9	7	774	40701	174573	5652372	273.29
140	2	8	7	760	35590	238184	7015157	334.43
Couverture $CCP1$ - nombre de jokers décrémenté de 1 entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	6	10	5	2	14	22	154	3.07
50	5	9	5	2	14	138	966	7.75
100	3	9	7	251	170	42463	20316	25.37
140	2	8	7	247	161	57802	27761	62.84

Tableau 5.5: Comportement de Combi pour la couverture $CCP1$ et la variante 5.4.2.2.6.1 toute seule ou combinée à la variante 5.4.2.2.6.3 et pour $q = 70\%$.

Couverture $CCP2$ - pas de décrémentation du nombre de jokers entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	1	10	10	4	3	30	21	1.69
50	2	9	8	516	4325	44121	253418	53.87
100	2	9	8	433	666	72454	82070	63.11
140	1	8	8	421	613	97335	110270	68.15
Couverture $CCP2$ - nombre de jokers décrémenté de 1 entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	1	10	10	4	3	30	21	1.69
50	3	9	7	1	1	50	48	7.04
100	3	9	7	251	4278	42463	571817	41.28
140	1	8	8	421	613	97335	110270	69.28

Tableau 5.6: Comportement de Combi pour la couverture $CCP2$ et la variante 5.4.2.2.6.1 toute seule ou combinée à la variante 5.4.2.2.6.3 et pour $q = 70\%$.

Couverture $CCP3$ - pas de décrémentation du nombre de jokers entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	4	10	7	1239	5	23787	37	14.78
50	4	9	6	586	43	88047	2466	46.04
100	4	9	6	559	11	167338	1260	81.09
140	3	8	6	550	5	229444	901	102.11
Couverture $CCP3$ - nombre de jokers décrémenté de 1 entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	6	10	5	2	4	22	44	3.10
50	5	9	5	2	4	138	276	8.48
100	5	9	5	53	7	9086	918	22.72
140	4	8	5	50	5	12062	1000	45.78

Tableau 5.7: Comportement de Combi pour la couverture $CCP3$ et la variante 5.4.2.2.6.1 toute seule combinée à la variante 5.4.2.2.6.3 et pour $q = 70\%$.

Couverture $CCP1_{card}$ - pas de décrém. du nombre de jokers entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	1	10	10	4	1	30	7	1.94
50	2	9	8	516	38	44121	2076	32.27
100	3	9	7	774	27393	174573	4025253	267.33
140	2	8	7	760	24337	238184	5043267	327.35
Couverture $CCP1_{card}$ - nombre de jokers décrémenté de 1 entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	1	10	10	4	1	30	7	1.93
50	3	9	7	1	1	50	48	7.49
100	3	9	7	251	657	42463	84459	30.88
140	2	8	7	247	558	57802	102640	80.00

Tableau 5.8: Comportement de Combi (variante 5.4.2.2.6.2) pour la couverture $CCP1_{card}$ et la variante 5.4.2.2.6.1 toute seule ou combinée à la variante 5.4.2.2.6.3 et pour $q = 70\%$.

Couverture $CCP2_{card}$ - pas de décrém. du nombre de jokers entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	1	8	8	2	4	16	28	0.36
50	2	7	6	159	640	13638	44709	5.47
100	1	6	6	126	363	21570	51206	6.63
140	1	6	6	126	299	29776	58895	8.96
Couverture $CCP2_{card}$ - nombre de jokers décrémenté de 1 entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	1	8	8	2	4	16	28	0.36
50	2	7	6	2	4	121	284	1.48
100	1	6	6	126	363	21570	51206	6.66
140	1	6	6	126	299	29776	58895	8.96

Tableau 5.9: Comportement de Combi (variante 5.4.2.2.6.2) pour la couverture $CCP2_{card}$ et la variante 5.4.2.2.6.1 toute seule ou combinée à la variante 5.4.2.2.6.3 et pour $q = 70\%$.

Couverture $CCP3_{card}$ - pas de décrém. du nombre de jokers entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	2	10	9	225	58	2182	431	5.28
50	2	9	8	516	1	44121	50	30.72
100	3	9	8	774	15867	174573	2237249	271.85
140	3	8	7	760	13324	238184	2662072	306.97
Couverture $CCP3_{card}$ - nombre de jokers décrémenté de 1 entre les 2 phases								
N	nombre de cycles	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
10	5	10	5	2	14	22	154	3.19
50	5	9	5	2	14	138	966	8.28
100	2	9	7	251	69	42463	8341	26.88
140	2	8	7	247	51	57802	8852	66.97

Tableau 5.10: Comportement de Combi (variante 5.4.2.2.6.2) pour la couverture $CCP3_{card}$ et la variante 5.4.2.2.6.1 toute seule ou combinée à la variante 5.4.2.2.6.3 et pour $q = 70\%$.

N	#tours	k'_{max}	k_{max}	#mod. phase 1	#mod. phase 2	#occ. phase 1	#occ. phase 2	temps (s)
4	1	17	17	9	32	18	64	1.72
6	3	14	13	264	894	47	141	6.26
8	3	14	12	218	1311	8	32	21.57
10	4	14	11	246	2133	125	625	113.33
12	3	13	11	200	2170	14	84	74.47
14	4	13	10	177	2375	137	1007	110.18
16	4	13	10	179	2867	111	942	119.21
18	4	13	10	162	2952	16	153	111.98
20	4	13	10	151	3136	4	41	103.67
22	4	13	10	128	3012	3	34	87.51
24	5	13	9	144	3718	332	4320	152.74
26	5	13	9	132	3795	180	2569	149.12
28	4	12	9	136	4203	134	2051	155.47
30	4	12	9	151	4946	340	5654	194.48

Tableau 5.11: Comportement de l'algorithme Combi original pour $q = 70\%$, de 2 à 30 chaînes de promoteurs et en utilisant la couverture CCP.

Enfin, les tableaux 5.11 et 5.12 présentent une comparaison des trois approches, directe, minimalité et esquisse de l'espace solution, de Combi. Le premier des deux tableaux indique simplement les valeurs des divers paramètres intervenant dans le comportement de la version finale de Combi (en particulier le nombre de fois où l'algorithme doit exécuter les deux phases). Le second tableau compare les temps d'exécution des trois approches. Nous pouvons observer que :

- la version finale est d'autant plus performante par rapport aux approches directe ou minimalité que la valeur de k'_{max} est proche de celle de k_{max} (i.e. lorsque l'algorithme doit effectuer peu de cycles), elle peut par contre être légèrement moins rapide que la version minimalité si ces deux valeurs sont éloignées (par exemple ici pour N égal à 24, 26, 28 ou 30);
- le rapport des temps d'exécution des versions directe et minimalité est pratiquement constant et approximativement égal à 4 (la version directe est 4 fois plus lente que la version minimalité).

5.4.2.3 Couverture pondérée et distance de Levenshtein

5.4.2.3.1 Introduction

Nous avons mentionné dans la section 5.4.2.2.6.4 qu'il n'est pas possible, à l'heure actuelle,

Nombre de séquences	Temps		
	Direct	Minimalité	Esquisse
4	7648.70	620.14	1.72
6	991.60	306.28	6.26
8	526.19	183.89	21.57
10	539.67	179.57	113.33
12	477.28	146.90	74.47
14	367.25	110.76	110.18
16	404.23	115.45	119.21
18	404.68	118.35	111.98
20	397.48	107.63	103.67
22	377.62	108.46	87.51
24	388.93	99.43	152.74
26	394.55	98.61	149.12
28	418.61	103.58	155.47
30	498.08	122.58	194.48

Tableau 5.12: Comportement de l'algorithme Combi original, approches 'directe' (section 5.4.2.2.3), 'minimalité' (section 5.4.2.2.4) et 'esquisse de l'espace solution' (section 5.4.2.2.5) pour $q = 70\%$, de 2 à 30 chaînes de promoteurs et en utilisant la couverture CCP.

de travailler avec une couverture totalement combinatoire lorsque nous voulons comparer des protéines. Nous avons indiqué alors une façon de procéder lorsque nous voulons essayer malgré tout d'analyser ces macromolécules sans point de vue a priori.

En fait, le sous-ensemble le plus intéressant de Σ correspond à $\{\Sigma\}$, qu'il s'agisse d'analyser des protéines ou des acides nucléiques. La section 3.5.2.1.2 a montré en effet qu'introduire des jokers dans les modèles permet de repérer des régions d'une macromolécule dont toutes les positions ne sont pas impliquées dans l'activité biologique. Par ailleurs, il peut être important en plus d'autoriser des erreurs entre les modèles et leurs instances ainsi que nous l'avons fait dans l'algorithme LePoivre. Nous allons donc considérer que nous disposons maintenant d'une couverture comportant un certain nombre de sous-ensembles de l'alphabet Σ dont le joker, et que ce dernier est le seul élément de la couverture qui ne reçoit pas une pondération infinie. Notons z la pondération de $\{\Sigma\}$. Observons que la couverture peut être très dégénérée et comporter des éléments dont certains sont strictement inclus dans d'autres. Elle peut également être établie sur la base de critères biologiques (structuraux par exemple) ou combinatoires du genre de celui suggéré dans la section 5.4.2.2.6.4.

Nous allons alors établir qu'un mot dans une chaîne est une occurrence d'un modèle défini sur une telle couverture (ce modèle ne possède pas plus de z jokers) si ce mot se trouve à une distance Ensembliste de Levenshtein au plus égale à e du modèle. Nous introduisons d'abord formellement le lemme permettant de construire de tels modèles, puis nous indiquons comment obtenir l'algorithme à partir de ceux pour une distance de Levenshtein et pour une couverture combinatoire pondérée. Pour simplifier, nous nous contentons en fait de 'compter les erreurs' pendant le déroulement de l'algorithme, ce qui veut dire que, par rapport à la distance Ensembliste de Levenshtein et au lemme 5.4.4 de LePoivre, nous travaillons ici avec l'ensemble $ES(M)$ plutôt qu'avec l'ensemble $EL(M)$ pour M un modèle. Les 'vraies' occurrences en termes d'une distance de Levenshtein peuvent être calculées à la fin de l'algorithme comme cela était déjà fait dans les versions de LePoivre présentées dans la section 5.4.2.1.5.

5.4.2.3.2 Définitions de récurrence

Nous commençons par définir une couverture combinatoire pondérée restreinte $CCPR$ de l'alphabet Σ des monomères.

Définition 5.4.3 *Une couverture combinatoire pondérée restreinte $CCPR$ de Σ est définie de la façon suivante :*

$$CCPR = \left\{ \begin{array}{l} (S, \infty)\ \forall S \in C \text{ où } C \text{ est un ensemble de sous} - \text{ensembles propres de } \Sigma, \\ (S, 0) \text{ pour } S \in \mathcal{P}^+(\Sigma) \setminus C, \\ (\{\Sigma\}, z) \text{ où } z \text{ est un entier } fini\ non - négatif \end{array} \right\}.$$

Dans le cas des acides nucléiques, nous avons en général $C = CI$, et dans celui des protéines C sera une couverture comme celles établie à partir du diagramme de Taylor [Taylor, 1986a] et donnée dans la figure 3.11 du chapitre 3.

Soient $s \in \Sigma^*$, i une position dans s, $CCPR$ une couverture combinatoire pondérée restreinte de Σ et $M_1 \in CCPR^k$. Nous avons alors le lemme suivant (rappelons que $Occ(M)$ désigne un ensemble un peu plus grand que celui des occurrences strictes d'un modèle M puisqu'il s'agit de l'ensemble de tous les mots ayant au plus $d \leq e$ erreurs avec M où d peut ne pas représenter une distance) :

Lemme 5.4.6

$$(i, d_s, d_d, d_i) \in Occ(M = M_1 S) \; avec \; S \in CCPR$$

$$\Longleftrightarrow$$

$$\exists \, d_{s_1}, d_{d_1}, d_{i_1} \; tel \; que \; (i, d_{s_1}, d_{d_1}, d_{i_1}) \in Occ(M_1) \; et \; :$$

$$[Condition \; 1 \; et \; (Condition \; 2 \; ou \; Condition \; 3)]$$

$$ou$$

$$[Condition \; 4]$$

avec :

Condition 1 $S \neq \{\Sigma\}$*;*

Condition 2 $s_i+ \mid M_1 \mid -d_{d_1} + d_{i_1} \in S$*;*

Condition 3 $s_i+ \mid M_1 \mid -d_{d_1} + d_{i_1} \notin S$ *et* $d_s + d_d + d_i = d_{s_1} + d_{d_1} + d_{i_1} + 1 \leq e$*;*

Condition 4 $S = \{\Sigma\}$ *et* $(p(S, M_1) + 1) \leq z$ *où* $p(S, M_1)$ *est le nombre de fois où l'ensemble* S *apparaît dans le modèle* M_1*.*

5.4.2.3.3 Algorithme

L'algorithme établi à partir du lemme et permettant de construire l'ensemble des occurrences d'un modèle M dans le sens d'une couverture combinatoire restreinte $CCPR$ est esquissé dans la figure 5.29. Pour les raisons indiquées dans la section 5.4.2.2.5.4, le joker est interdit en première ainsi qu'en dernière position d'un modèle.

/* Entrée */
$CCPR$: une couverture combinatoire pondérée restreinte établie à partir d'une couverture C
p : le nombre d'éléments dans C
e, z : le nombre d'erreurs et de jokers autorisés respectivement
Le reste comme pour les algorithmes précédents
/* Sortie */
Comme pour les algorithmes précédents
/* Principales structures de données */
Comme pour les algorithmes des figures 5.15 et 5.28, sauf le tableau Modèles[i] qui n'est pas utilisé

```
Main {
    pour (i ∈ [1..n])
        pour (S_j ∈ C)
            si (i ∈ S_j)
                Occ(S_j) = Occ(S_j) ∪ {(i,0,0,0)};
            sinon
                Occ(S_j) = Occ(S_j) ∪ {(i,1,0,0)} ∪ {(i,0,1,0)}};
    pour (S_j ∈ C)
        si (Occ(S_j) vérifie le quorum q)
            Explore(S_j, 1, Occ(S_j));
}
```

Figure 5.29 : Algorithme LeCombi (couverture combinatoire pondérée restreinte et distance de Levenshtein — ici simple comptage des erreurs) pour résoudre le Problème 1 — continue à la page suivante.

5.4.2.3.3.1 Complexité

Il est aisé de déduire de la complexité de LePoivre que la complexité en temps de LeCombi est égale à (opération de filtrage exclue) $O(n.N.(e+2).(2e+1).k^{e+1}.(p+1)^{e+1}.2^k)$ où p est le nombre d'éléments dans C. Dans le cas de l'ADN et de l'ARN en particulier, il est courant d'avoir $p = \mid \Sigma \mid$. La complexité en espace de l'algorithme est bornée supérieurement par $O(n.N.k.(2e+1))$ pour un parcours en profondeur.

5.4.2.3.4 Amélioration

Il est possible d'appliquer ici les mêmes modifications que pour l'algorithme LePoivre, en particulier celle de la section 5.4.2.1.5.

La complexité de l'algorithme est alors $O(n.N.(e+1).(e+2).k^{e+1}.(p+1)^{e+1}.2^k + k.(k+e).N_{ModèlesValides})$ en temps et $O(n.N.k)$ en espace.

C'est cette version améliorée dont nous analysons la performance en pratique dans la prochaine section.

5.4.2.3.5 Performance en pratique

Nous présentons dans cette section la performance de LeCombi lorsque nous faisons varier le

```
Explore(M, l, Occ(M)) {
    si (l = k)                                  /* base de la récursion */
        si (le joker n'est pas le dernier élément de M)
            stocke M et Occ(M);
    sinon {                                     /* pas de la récursion */
        pour ((i, d_s, d_d, d_i) ∈ Occ(M)) {
            pour (S = ∅)                        /* insertion */
                si (d_s + d_d + d_i + 1 ≤ e) {
                    Occ(MS) = Occ(MS) ∪ {(i, d_s, d_d, d_i + 1)};
                    ExtensionPossible = ExtensionPossible ∪ {S};
                }
            pour (S = {Σ})
                si ((p_M[{Σ}] + 1) ≤ z) {
                    Occ(MS) = Occ(MS) ∪ {(i, d_s, d_d, d_i)};
                    ExtensionPossible = ExtensionPossible ∪ {S};
                }
            pour (S ∈ C)
                si (s_i + l − d_d + d_i ∈ S) {
                                            /* identité */
                    Occ(MS) = Occ(MS) ∪ {(i, d_s, d_d, d_i)};
                    ExtensionPossible = ExtensionPossible ∪ {S};
                }
                sinon si (d_s + d_d + d_i + 1 ≤ e) {
                                            /* substitution ou délétion */
                    Occ(MS) = Occ(MS) ∪ {(i, d_s + 1, d_d, d_i)};
                    Occ(MS) = Occ(MS) ∪ {(i, d_s, d_d + 1, d_i)};
                    ExtensionPossible = ExtensionPossible ∪ {S};
                }
        }
        pour (S ∈ ExtensionPossible)
            si (Occ(MS) vérifie le quorum q)
                Explore(MS, l + | S |, Occ(MS));
    }
}
```

Figure 5.29: Algorithme LeCombi (couverture combinatoire pondérée restreinte et distance de Levenshtein — ici simple comptage des erreurs) pour résoudre le Problème 1.

Nombre de jokers	0 erreur		1 substitution		1 erreur	
	k_{max}	temps (s)	k_{max}	temps (s)	k_{max}	temps (s)
0	4	0.076	6	1.03	7	2.10
1	5	0.19	7	4.47	8	9.95
2	6	0.37	9	12.51	9	29.94
3	7	0.68	10	28.34	10	70.94
4	8	0.94	11	56.78	11	145.98
5	9	1.36	12	101.69	12	274.72
6	10	1.86	13	171.31	13	491.43
7	11	2.49	14	275.70	15	1323.78
8	12	3.15	15	420.75	15	2065.43
9	13	3.96	16	617.61	16	3115.67
10	14	4.89	17	963.95	18	2769.30
11	15	5.95	18	1233.85	19	3939.74
12	16	7.17	19	1689.71	20	5524.03

Tableau 5.13: Comportement de LeCombi (longueur maximale des modèles trouvés et temps d'exécution en secondes) pour la couverture CCP donnée dans le texte, $q = 100\%$, $N = 20$, $k = k_{max}$, un nombre de jokers variant entre 0 et 12, et 0 erreur, 1 substitution et 1 erreur (substitution ou trou).

nombre de jokers autorisés. Les modèles recherchés étaient les plus longs présents dans toutes les chaînes d'un ensemble réduit (20 chaînes) du jeu d'essai composé des promoteurs de *Bacillus subtilis* utilisé dans la section 5.4.2.1.7. Enfin, nous avons autorisé 0 erreur, 1 substitution ou 1 erreur au plus. La couverture employée est dans tous les cas la suivante :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{A, C, G, T\}, J), \\ (S, 0)\ \textit{pour tout autre ensemble} \end{array} \right\}$$

avec J (nombre de jokers) variant entre 0 et 12.

Le tableau 5.13 indique les temps d'exécution obtenus sur une Sparc Station 20, ainsi que les valeurs des longueurs maximales des modèles. La figure 5.30 présente les courbes correspondantes, montrant l'évolution du temps d'exécution en fonction du nombre de jokers. Ces courbes ne sont pas linéaires mais il faut observer qu'ici la longueur maximale varie avec le nombre de jokers autorisés. Nous n'avons donc pas le même comportement avec LeCombi qu'avec H/LeMoivre ou H/LePoivre (voir section 5.4.2.1.6.2) pour lesquels la longueur maximale des modèles décroissait assez rapidement au début mais devenait vite ensuite pratiquement constante.

Figure 5.30: Performance de LeCombi : temps (en secondes) $\times$ nombre de jokers pour la couverture CCP donnée dans le texte, $q = 100\%$, $N = 20$, $k = k_{max}$ et 0 erreur, 1 substitution et 1 erreur (substitution ou trou) (courbes 1, 2 et 3 respectivement).

5.4.3 L'algorithme utilisant des mots comme unités de comparaison

5.4.3.1 Introduction

Le dernier algorithme que nous présentons travaille avec la définition de ressemblance 3.5.15. Il a été introduit dans [Sagot *et al.*, 1995c] et emploie une notion de similarité entre mots dans les chaînes et modèles (qui, dans ce cas, sont eux-mêmes des mots) qui repose sur les sous-mots de ces deux objets. Nous avons vu dans la section 3.2.4.1.1 du chapitre 3 que cette mesure peut souvent être plus flexible que celle basée sur la distance de Levenshtein utilisée par LePoivre (section 5.4.2.1.4).

L'algorithme est présenté progressivement. Nous commençons par la version sans trou, puis nous introduisons une version simplifiée de l'algorithme avec trous. Dans la définition 3.5.16, cette version consiste à ne permettre qu'au plus un seul trou par sous-mot de longueur w des modèles. Nous indiquons finalement comment peut être traité le cas où plus d'un trou est autorisé par sous-mot.

Nous allons voir que, dans tous les cas, l'algorithme construit d'abord les modèles de longueur w (ses unités de comparaison) ainsi que leurs ensembles d'occurrences. Dans un premier temps, nous n'allons pas nous soucier de comment ni où ces objets sont stockés, et allons donc simplement supposer qu'ils sont maintenus dans une table indexée. Nous montrerons ensuite quelles autres structures il est possible d'utiliser.

5.4.3.2 Version sans trou

5.4.3.2.1 Définition de récurrence

Soient s une chaîne de longueur n, $\mathcal{M}$ une matrice (que nous supposerons de similarité), $m = m_1...m_k$ un modèle, $i \in \{1,...,n- \mid m \mid +1\}$ une position dans s et $u = u_1...u_k$ le mot de longueur k commençant à la position i dans s. Alors le lemme suivant est vérifié :

Lemme 5.4.7 *Pour tout* $(k-1) \geq w$*:*

$$(i, u = u_1...u_k) \in K(m = m_1...m_k)$$

$$\Longleftrightarrow$$

$$(i, u_1...u_{k-1}) \in K(m_1...m_{k-1})\ et$$

$$l_w(u, m) \geq t.$$

Rappelons que $l_w(u, m)$ indique le score des derniers sous-mots de longueur w de u et m. Ce lemme nous fournit une propriété constructive permettant d'obtenir $K(m_1...m_{k-1}m_k)$ à partir de $K(m_1...m_{k-1})$ pour tout $(k-1) \geq w$. Nous allons faire l'hypothèse pour l'instant que tous les modèles de longueur w qui sont w-présents dans s ont déjà été trouvés et que nous connaissons leurs ensembles d'occurrences. Nous discutons plus loin comment déterminer ces objets (section 5.4.3.2.2). Avant d'indiquer la propriété, observons que la valeur de $l_w(u, m)$ peut être calculée en $O(1)$ opérations puisque :

$$l_w(u, m) = l_w(u_1...u_{k-1}, m_1...m_{k-1}) - \mathcal{M}(u_{k-w}, m_{k-w}) + \mathcal{M}(u_k, m_k)$$

sachant que :

$$l_w(u_1...u_{k-1}, m_1...m_{k-1}) = score_{\mathcal{M}}(u_1...u_{k-1}, m_1...m_{k-1})\ si\ (k-1) = w.$$

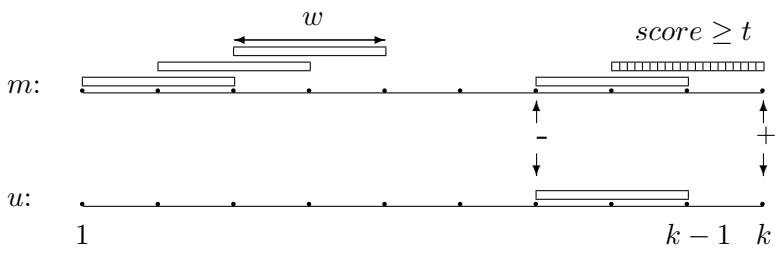


Figure 5.31: Extension d'un modèle.

Proposition 5.4.4 *Étant donné un ensemble d'occurrences* $K(m = m_1...m_{k-1})$ *avec* $(k-1) \geq w$*, nous avons :*

$$K(m = m_1...m_k)$$
$$=$$
$$\{(i, u = u_1...u_k) \mid (i, u_1...u_{k-1}) \in K(m_1...m_{k-1}) \text{ et } l_w(u, m) \geq t\}.$$

Si pour chaque élément (i, u) dans $K(m)$ nous conservons le score du dernier mot de longueur w de u et de m, alors nous pouvons construire les occurrences de n'importe quel modèle m connaissant uniquement les occurrences de ses préfixes de longueur $\mid m \mid$ - 1 (voir la figure 5.31).

Nous montrerons plus loin que si les valeurs de $\mathcal{M}(a, b)$ sont ordonnées pour tout $a \in \Sigma$, nous pouvons économiser un certain nombre d'opérations, à la fois au moment de la construction des modèles de longueur w et à celui de leur allongement.

5.4.3.2.2 Algorithme

L'idée de l'algorithme est donné dans la figure 5.33. Celui-ci s'appelle Klast (de Kmr + bLAST) et fonctionne en deux étapes distinctes : une étape d'initialisation et une étape d'extension. L'étape d'initialisation consiste à rechercher de manière efficace tous les modèles de longueur w qui sont w-présents dans au moins q des chaînes $s1, s2, ..., sN$. La construction de ces modèles n'a pas besoin d'être réalisée d'une manière complètement naïve, c'est-à-dire, nous n'avons pas besoin de vérifier pour tous les modèles possibles de longueur w s'ils ont des occurrences dans $s = s1s2...sN$. Une simple observation nous permet de limiter l'espace de recherche à cette étape. Celle-ci concerne le fait qu'un mot u de s ne peut être une w-occurrence d'un modèle m s'il existe $p \leq w$ tel que :

$$score_{\mathcal{M}}(u_1...u_p, m_1...m_p) + (w - p) * max(\mathcal{M}) < t$$

où $max(\mathcal{M})$ est la valeur maximale de la matrice de similarité $\mathcal{M}$.

Dans l'étape d'extension, nous considérons chaque modèle de longueur w, appelons m ce modèle, et nous essayons de l'allonger le plus loin possible, c'est-à-dire, tant que m a des occurrences dans au moins q des chaînes de l'ensemble. Cette seconde étape repose sur le lemme 5.4.7.

Soit $\Sigma = \{a, b, c, d, e\}$ et soit $\mathcal{M}$ la matrice suivante :

	a	b	c	d	e
a	5	3	0	2	1
b	3	4	1	2	4
c	0	1	5	4	2
d	2	2	4	4	1
e	1	4	2	1	4

Alors la matrice d'ordre $\mathcal{O}$ de $\mathcal{M}$ est la matrice donnée par :

	1	2	3	4	5
a	a	b	d	e	c
b	b	e	a	d	c
c	c	d	e	b	a
d	d	c	a	b	e
e	e	b	c	a	d

Figure 5.32: Exemple d'une matrice d'ordre.

Pendant les deux phases détaillées dans la figure 5.33, l'algorithme fait usage d'une matrice, appelée 'matrice d'ordre', qui est liée à $\mathcal{M}$ de la façon suivante :

Notation 5.4.6 *Soit une matrice $\mathcal{M}$, nous notons $\mathcal{O}$ la matrice d'ordre de $\mathcal{M}$ définie sur* $\mid \Sigma \mid \times \mid \Sigma \mid$ *par :*

$$\forall\ a \in \Sigma, \forall\ j \in [1..(\mid \Sigma \mid - 1)],\ \mathcal{M}(a, \mathcal{O}(a, j)) \leq \mathcal{M}(a, \mathcal{O}(a, j+1)).$$

Chaque rang de la matrice $\mathcal{O}$ correspond ainsi à un symbole de Σ, et présente les symboles de Σ par scores, avec ce symbole, décroissants (voir la figure 5.32).

5.4.3.2.3 Complexité

Si les modèles de longueur w ainsi que les positions de leurs occurrences sont conservés dans une table indexée (temps d'accès en $O(1)$), la complexité en temps de l'algorithme est au pire en $O(n.N. \mid \Sigma \mid^w)$ pour la première étape. Celle de la seconde est $O(n.N.N_{MaxBranches}.k.N_{Modi}(k-1))$ où $N_{MaxBranches} = \mid \Sigma \mid$. Dans le cas de Klast, $N_{Modi}(k-1)$ va en pratique dépendre fortement de plusieurs facteurs : la chaîne s, la matrice utilisée, le choix du seuil t et du quorum q. Le calcul de sa valeur théorique est une question qui demeure en ouvert. La complexité globale en temps est donc $O(n.N. \mid \Sigma \mid^w) + O(n.N. \mid \Sigma \mid .k.N_{Modi}(k-1))$. Une observation importante cependant est que $N_{Modi}(k-1)$ ne dépend pas directement de n, bien que la contribution de $N_{Modi}(k-1)$ à la complexité en temps de l'algorithme puisse être grande si t est fixé à une petite valeur et $N_{Modi}(k-1)$ peut avoir alors une relation avec $n.N$.

La complexité en espace de l'algorithme est $O(max\{n.N. \mid \Sigma \mid^w, n.N.N_{Modi}(k-1)\})$ si les modèles de longueur w ainsi que leurs occurrences sont stockées dans une table indexée.

```
/* Entrée */
        s = s1s2...sN : une chaîne de longueur n = concaténation de N chaînes
        M : une matrice de similarité
        O : matrice d'ordre de M, comme dans la notation 5.4.6
        t, w > 0, q : seuil, longueur des sous-mots et contrainte de quorum respectivement
        k : une longueur fixée
/* Sortie */
        Tous les modèles m de longueur k ayant des occurrences dans au moins q chaînes
/* Principales structures de données */
        K(m) = ensemble des occurrences d'un modèle m - contient pour chaque occurrence
        non seulement sa position i et l'image correspondante u mais aussi la valeur
        l_w(u, m) notée plus simplement l_w : implémenté comme une pile
        ExtensionPossible = ensemble des modèles qui vérifient le quorum q et peuvent
        donc être étendus : implémenté comme une pile
        T(m) = ensemble des occurrences du modèle m de longueur w : implémenté
        comme une pile
        E = ensemble des modèles de longueur w qui ont au moins une occurrence dans s
Main {
        pour (i ∈ [1..(n − w + 1)])
                Énumère (i, i − 1, λ, 0, O, T, E);
        pour (m ∈ E)
                si (T(m) ≠ ∅)
                        si (T(m) vérifie le quorum q)
                                Explore (m, w, T(m));
}

/* Étape d'initialisation */
/* d et f sont les positions de début et fin respectivement du sous-mot de longueur */
/* l de s, et m est le modèle en construction                                        */
Énumère (d, f, m, l, O, T, E) {
        si (l = w)                                          /* base de la récursion */
                si (score_M (s_d...s_f, m) ≥ t) {
                        T(m) = T(m) ∪ {(d, s_d...s_f);
                        E = E ∪ {m};
                }
        sinon {                                             /* pas de la récursion */
                si ((l > 0) et (score_M (s_d...s_f, m) + (max(M) * (w − l)) < t))
                        retourner;                          /* élagage */
                pour (j ∈ [1.. | Σ |]) {
                        a = O(s_{f+1}, j);
                        Énumère (d, f + 1, ma, l + 1, O, T, E);
                }
}
```

Figure 5.33 : Algorithme Klast (distance ayant pour base des mots, pas de trou) pour résoudre le Problème 1 — continue à la page suivante.

```
/* Étape d'extension */
Explore(m = m_1...m_l, l, K(m)) {
        si (l = k)                                          /* base de la recursion */
                stocke m et K(m);
        sinon {                                             /* pas de la recursion */
                pour ((i, u = s_i...s_{i+l-1}, l_w) ∈ K(m))
                        pour (j ∈ 1.. | Σ |) {
                                a = O(s_{i+l}, j);
                                l'_w = l_w − M(s_i, m_1) + M(s_{i+l}, a);
                                si (l'_w ≥ t) {
                                        K(ma) = K(ma) ∪ {(i,us_{i+l}, l'_w)};
                                        ExtensionPossible = ExtensionPossible ∪ {ma};
                                }
                                sinon
                                        sort de la boucle 'pour j';
                        }
                pour (ma ∈ ExtensionPossible)
                        si (K(ma) vérifie le quorum q)
                                Explore(ma, l + 1, K(ma));
        }
}
```

Figure 5.33: Algorithme Klast (distance ayant pour base des mots, pas de trous) pour résoudre le Problème 1.

5.4.3.3 Version avec trou

5.4.3.3.1 Esquisse de l'algorithme

Nous esquissons maintenant une extension de Klast qui autorise des trous (insertions et délétions) entre les modèles et leurs occurrences. Les notations sont celles de la définition 3.5.16. Dans l'implémentation actuelle, la distance entre deux trous ne peut être plus petite que w (c'est-à-dire, $e_s = 1$) et le score pour aligner un trou avec un symbole est une constante fixée, pour simplifier, à zéro. De plus, les trous ne sont pas autorisés dans le premier sous-mot. Un certain nombre de variables supplémentaires sont maintenant nécessaires. Pour chaque w-occurrence $(i, u_1...u_{k-1})$ d'un modèle $m_1...m_{k-1}$, une variable p_s indique ainsi la distance du dernier trou présent dans l'alignement du modèle et de l'occurrence et une variable p_t indique le nombre total de trous. Lorsqu'un modèle est étendu d'un symbole, p_s est incrémenté et un trou est autorisé entre le modèle m étendu et l'occurrence correspondante u également étendue seulement si $l_w(u, m) \geq t, p_s > w$ et $p_t < e_t$. L'idée de l'algorithme est donnée dans la figure 5.34.

Dans la section 5.4.3.2.1, nous avions indiqué que la valeur de $l(u = u_1...u_k, m = m_1...m_k)$ pouvait être calculée simplement à partir de celle de $l(u_1...u_{k-1}, m_1...m_{k-1})$ pour $(k-1) \geq w$ en $O(1)$ opérations. Ce calcul demeure constant ici, cependant la formule permettant de passer d'une valeur à l'autre fait maintenant intervenir le type de trou introduit en dernier lieu dans l'alignement de $u_1...u_{k-1}$ et $m_1...m_{k-1}$. Nous avons ainsi :

- si $p_s \geq w$:
  $l_w(u, m) = l_w(u_1...u_{k-1}, m_1...m_{k-1}) - \mathcal{M}(u_{k-w}, m_{k-w}) + \mathcal{M}(u_k, m_k)$;
- si $p_s < w$:
  - si le dernier trou introduit était une insertion :
    $l_w(u, m) = l_w(u_1...u_{k-1}, m_1...m_{k-1}) - \mathcal{M}(u_{k-w-1}, m_{k-w}) + \mathcal{M}(u_k, m_k)$;
  - si le dernier trou introduit était une délétion :
    $l_w(u, m) = l_w(u_1...u_{k-1}, m_1...m_{k-1}) - \mathcal{M}(u_{k-w+1}, m_{k-w}) + \mathcal{M}(u_k, m_k)$.

Au niveau de l'algorithme, il faut donc conserver une information de plus pour chaque occurrence, qui concerne justement le type du dernier trou introduit.

L'extension à plus d'un trou par sous-mot va exiger non plus deux valeurs indiquant respectivement la position du dernier trou (p_s) et son type, mais deux tableaux de bits (c'est-à-dire en fait toujours deux entiers) dont les indices i indiquent si une erreur (délétion ou insertion) a été introduite dans u face à la position m_{k-i+1} de m et son type (par exemple bit à 0 pour une délétion et à 1 pour une insertion).

5.4.3.3.2 Complexité

La complexité en temps de l'algorithme pour la première étape est au pire en $O(n.N. \mid \Sigma \mid^w)$ comme précédemment. Celle de la seconde est $O(n.N.N_{MaxBranches}.k.F_{MultiPos}.N_{Modi}(k-1))$ où $N_{MaxBranches}$ est égal à $(e_s + 2) \mid \Sigma \mid$ et $F_{MultiPos}$ est égal à $(2e_t + 1)$. La complexité globale est donc $O(n.N. \mid \Sigma \mid^w) + O(n.N.(e_s + 2).(2e_t + 1). \mid \Sigma \mid .k.N_{Modi}(k-1))$.

La complexité en espace de l'algorithme est $O(max\{n.N. \mid \Sigma \mid^w, n.N.(2e_t + 1).N_{Modi}(k-1)\})$.

```
/* Entrée */
        e_t, e_s : nombre total de trous permis et distance minimale entre deux trous
        Le reste comme pour Klast
/* Sortie */
        Comme pour Klast
/* Principales structures de données */
        K(m) = ensemble des occurrences d'un modèle m - contient les mêmes informations
        que pour Klast avec de plus l'indication du nombre p_t de trous déjà introduits
        dans le mot, la distance p_s du dernier trou (initialement p_s = +∞), et une variable
        booléenne type indiquant le type de trou introduit (éventuellement) dans le dernier
        sous-mot
        Le reste comme pour Klast
/* Fonction principale : comme pour Klast */
/* Étape d'extension */
Explore(m = m_1...m_l, l, K(m)) {
        si (l = k)                                          /* base de la recursion */
                stocke m et K(m);
        sinon {                                             /* pas de la recursion */
                pour ((i, u = s_i...s_{i+l-1}, l_w, p_t, p_s) ∈ K(m))
                        pour (j ∈ 1.. | Σ |) {
                                p_s = p_s + 1;
                                a = O(s_{i+l}, j);
                                si (p_s > w)
                                        l'_w = l_w − M(s_i, m_1) + M(s_{i+l}, a);
                                sinon si (type = insertion)
                                        l'_w = l_w − M(s_{i−1}, m_1) + M(s_{i+l}, a);
                                        sinon si (type = délétion)
                                                l'_w = l_w − M(s_{i+1}, m_1) + M(s_{i+l}, a);
                                si (l'_w ≥ t) {
                                        K(ma) = K(ma) ∪ {(i,us_{i+l}, l'_w, p_t, 0)};
                                        ExtensionPossible = ExtensionPossible ∪ {ma};
                                }
                                sinon si (p_t < e_t et p_s > w) {
                                        K(ma) = K(ma) ∪ {(i,us_{i+l}, l'_w, p_t + 1, 0)};
                                        ExtensionPossible = ExtensionPossible ∪ {ma};
                                }
                        }
                pour (ma ∈ ExtensionPossible)
                        si (K(ma) vérifie le quorum q)
                                Explore(ma, l + 1, K(ma));
        }
}
```

Figure 5.34: Algorithme GKlast (distance ayant pour base des mots, trous autorisés) pour résoudre le Problème 1.

5.4.3.4 Conservation des modèles initiaux

Dans ce qui précède, nous avons considéré que les modèles initiaux (ceux de longueur w) sont conservés dans une table indexée. D'autres structures de données sont bien sûr possibles. Nous avons ainsi réalisé une implémentation où ces modèles sont rangés dans un arbre AVL. Chaque nœud de l'arbre représente un entier codant pour le modèle et pointe sur une liste chaînée des positions où ce modèle est présent dans s. Comme pour la table indexée, la construction de l'arbre se fait en parcourant s et en plaçant la position i de chaque mot u de longueur w dans la liste de tous les modèles m de même longueur dont u est une occurrence. La recherche de m dans l'arbre prend à chaque fois dans le pire des cas un temps proportionnel au logarithme du nombre de modèles qui y ont été placés auparavant. L'algorithme est dans ce cas plus lent que la version utilisant une table, mais la structure en arbre a l'avantage de ne pas comporter plus de cellules qu'il n'y a de modèles de longueur w. Nous aurions pu également employer une table de hachage qui aurait représenté un compromis entre table indexée et arbre en termes d'espace en mémoire et rapidité de calculs. En pratique cependant, surtout lorsque nous souhaitons demeurer souples dans la définition de la similarité entre objets, presque tous les modèles de longueur w possèdent au moins une occurrence dans s, et ainsi pratiquement toutes les cellules de la table indexée se retrouvent remplies. C'est donc la structure que nous avons adopté en définitive.

5.4.3.5 Variantes : introduction de contraintes supplémentaires

5.4.3.5.1 Présence exacte (par sous-mots) des modèles au moins une fois

Comme nous l'avons mentionné, la complexité en temps de l'algorithme, quelle que soit la version, dépend du nombre maximal de modèles d'une certaine longueur qui se trouvent w-présents dans s. Ce nombre peut être réduit de manière importante si, en plus d'exiger qu'un modèle m soit w-présent au moins q fois dans s, nous imposons la contrainte supplémentaire que chaque sous-mot de longueur w de m soit exactement présent au moins une fois dans les chaînes (ce ne sera pas toujours nécessairement la même). Cela signifie que les modèles sont en fait construits à partir de sous-mots dans les chaînes.

Soit $EK(m)$ l'ensemble de w-occurrences de m dans le sens de la définition 3.5.15 de ressemblance. Alors :

$$EK(m) = \begin{cases} K(m) & si\ \forall j \in \{1, ..., \mid m \mid -w+1\},\ \exists u \in K(m)\ tel\ que \\ & score_M(u_j...u_{j+|m|-w+1}, m_j...m_{j+|m|-w+1}) \geq t \\ & et\ u_j...u_{j+|m|-w+1} = m_j...m_{j+|m|-w+1}; \\ \emptyset & autrement. \end{cases}$$

Il est important d'observer qu'un modèle m peut ne jamais être lui-même exactement présent dans s, puisque la constrainte ne concerne que ses sous-mots de longueur w. La modification à apporter à l'algorithme afin de respecter cette contrainte de présence se fait de manière très similaire à celle indiquée pour la seconde variante de H/LePoivre (section 5.4.2.1.6.2).

5.4.3.5.2 Seuil sur la moyenne de tous les sous-mots

Il est possible d'assouplir enocre la contrainte imposée au score des sous-mots des modèles avec les sous-mots des mots dans les chaînes. Au lieu d'exiger que chacun de ces scores soit supérieur ou égal à une certaine valeur seuil t, nous pouvons définir qu'un mot u dans une

chaîne est une occurrence d'un modèle m si et seulement si la moyenne des scores de chacun de ses sous-mots avec le sous-mot correspondant de m est supérieur ou égal à t. Plus formellement :

Lemme 5.4.8 *Pour tout* $(k-1) \geq w$*:*

$$(i, u_1...u_k) \in K(m_1...m_k)$$

$$\Longleftrightarrow$$

$$(i, u = u_1...u_{k-1}) \in K(m = m_1...m_{k-1}) \; et$$

$$\frac{\sum_{i=1}^{k} l_w(u_1...u_i + w - 1, m_1...m_{i+w-1})}{k-w+1} = \frac{\sum_{i=1}^{k} score(u_i...u_i + w - 1, m_i...m_{i+w-1})}{k-w+1} \geq t.$$

La modification à apporter à l'algorithme est simple : il suffit pour chaque modèle m de conserver une variable supplémentaire indiquant la moyenne obtenue jusqu'alors.

Observons que cette variante de l'algorithme initial est celle qui s'est révélée la plus utile en pratique ainsi que nous le verrons dans le chapitre consacré aux illustrations.

5.4.3.6 Performance en pratique

Nous présentons dans cette section les performances de Klast (original et sa variante travaillant avec un seuil établi sur la moyenne des scores des sous-mots — voir section 5.4.3.5.2) et de GKlast.

Comme nous l'avons mentionné, le temps d'exécution de l'algorithme dépend des données, de la matrice de similarité utilisée, de la valeur du seuil t et de celle du quorum q d'une façon que nous ne sommes pas capable d'établir de manière précise pour l'instant. Il est donc encore plus important dans ce cas d'évaluer le comportement de l'algorithme en pratique sur un 'vrai' ensemble test que pour les algorithmes présentés précédemment. Dans ce but, nous avons choisi comme jeu d'essai 80 protéines appartenant cette fois à la famille des facteurs d'élongation. Nous savons que les membres de cette famille possèdent plusieurs régions bien conservées (par exemple des sites de fixation de GTP). Par ailleurs, la longueur de ces protéines est assez homogène (environ 540 acides aminés).

Les figures 5.35 et 5.36 indiquent les courbes du temps d'exécution de Klast (courbes 1), Klast dans sa variante 'seuil sur la moyenne des sous-mots' (courbes 2) et GKlast (courbes 3) pour les valeurs suivantes de paramètres :

- matrice :

  Courbes a PAM250 normalisée (voir figure 3.7)

  Courbes b BLOSUM62 [Henikoff and Henikoff, 1992] (voir section 3.2.2.2)

- longueur des sous-mots : $w = 3$
- seuil : $t = 210$
- longueur des modèles recherchés : $k = 5$
- quorum : $q = 70\%$
- nombre de trous autorisés (pour GKlast) : $e_s = 1$ (1 trou au plus par sous-mot de longueur w)

Figure 5.35: Performance de Klast (original et variante 5.4.3.5.2) et GKlast (courbes 1, 2 et 3 respectivement) : temps (en secondes) $\times$ nombre de chaînes pour $w = 3$, $t = 210$, $k = 5$, $q = 70\%$, $e_s = 1$ (GKlast) et la matrice PAM250 et BLOSUM62 (courbes a et b respectivement).

Figure 5.36: Performance de Klast (original et variante 5.4.3.5.2) et GKlast (courbes 1, 2 et 3 respectivement) : temps (en secondes) $\times$ nombre de chaînes pour $w = 3$, $t = 210$, $k = 5$, $q = 70\%$, $e_s = 1$ (GKlast) et la matrice PAM250 et BLOSUM62 (courbes a et b respectivement) lorsque la présence exacte des modèles (par sous-mots) est exigée au moins une fois.

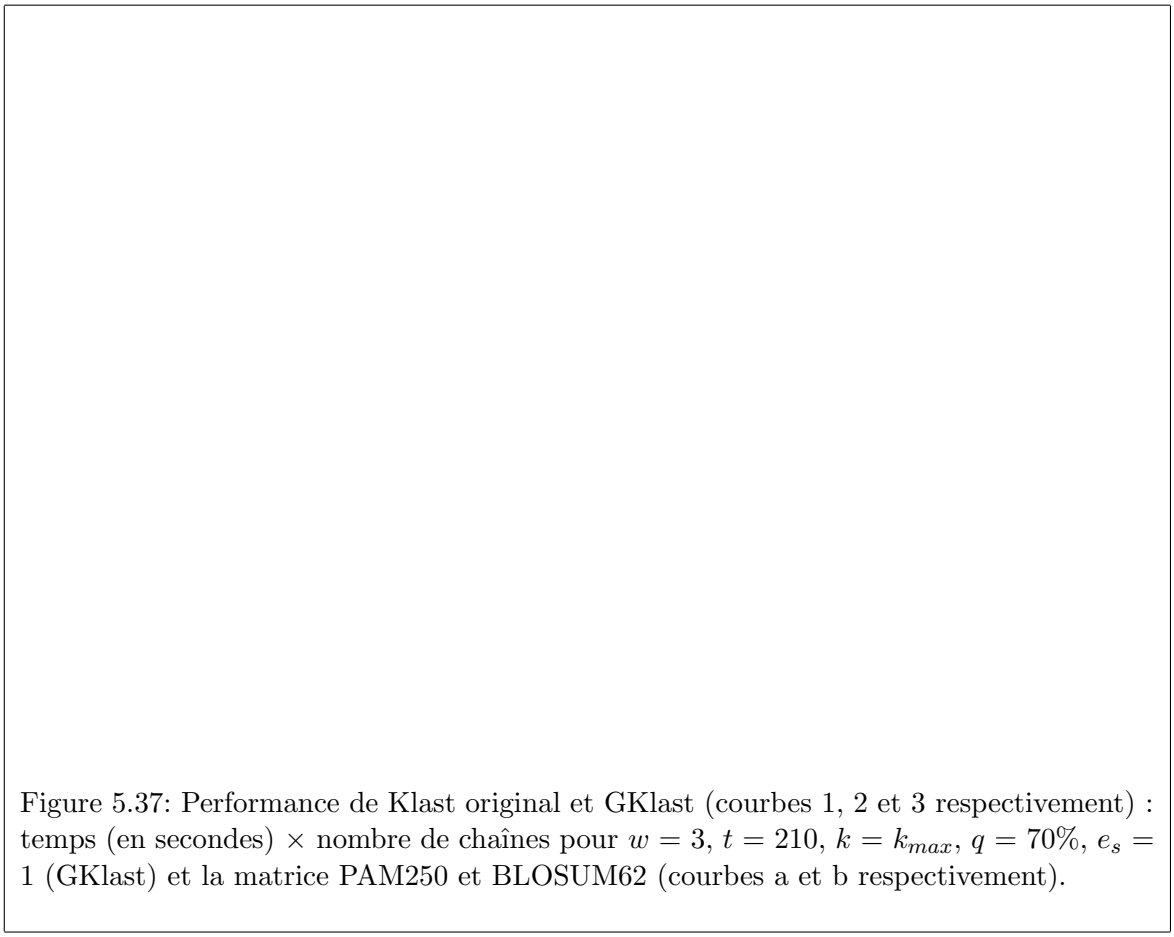


Figure 5.37: Performance de Klast original et GKlast (courbes 1, 2 et 3 respectivement) : temps (en secondes) $\times$ nombre de chaînes pour $w = 3$, $t = 210$, $k = k_{max}$, $q = 70\%$, $e_s = 1$ (GKlast) et la matrice PAM250 et BLOSUM62 (courbes a et b respectivement).

Figure 5.38: Performance de Klast (original et variante 5.4.3.5.2) et GKlast (courbes 1, 2 et 3 respectivement) : temps (en secondes) $\times$ quorum pour $w = 3$, $t = 210$, $k = 5$, $N = 20$, $e_s = 1$ (GKlast) et la matrice PAM250.

et lorsque le nombre de chaînes varie entre 5 et 80 avec un pas de 5. Dans la figure 5.36, nous exigeons de plus la présence exacte du modèle (par sous-mots) au moins une fois dans les chaînes de l'ensemble (variante de la section 5.4.3.5.1).

Nous pouvons déjà observer sur ces premières courbes que :

- quelle que soit la version de l'algorithme employé, sa complexité est bien en pratique linéaire en N, et donc en la longueur totale des chaînes puisque celles-ci possèdent approximativement le même nombre de résidus;
- le temps d'exécution est d'autant plus long que la matrice de similarité est plus dégénérée pour une même valeur de seuil (dans ce cas, la matrice la plus 'molle' est BLOSUM62);
- exiger la présence exacte du modèle (par sous-mots) au moins une fois dans les chaînes permet d'accélérer le temps de construction des modèles, mais le gain obtenu est relativement faible (environ 20 à 25%);
- comme nous pouvions nous y attendre GKlast est plus lent que Klast;
- la moins rapide des trois versions est celle qui travaille avec un seuil sur la moyenne des sous-mots (variante 5.4.3.5.2).

L'explication de ce dernier point est qu'un plus grand nombre de modèles est produit lorsque le seuil est établi sur la moyenne des scores de tous les sous-mots plutôt que sur celui du dernier (jusqu'à 11 fois plus de modèles peuvent être obtenus dans nos exemples). L'algorithme doit donc manipuler une masse plus importante d'objets durant toute son exécution.

Lorsque les modèles de longueur maximale sont recherchés, ceux-ci sont également plus longs, aussi bien pour cette variante de Klast que pour GKlast. Dans le cas de la variante, le rapport des longueurs par rapport à la version originale est en moyenne de 2,5 pour PAM250. Le temps d'exécution est quant à lui environ 44 fois plus grand. Ces rapports sont de 2 et 34 pour GKlast relativement à Klast. La figure 5.37 donne les courbes du temps d'exécution de Klast (courbes 1) et GKlast (courbes 2) en fonction des mêmes valeurs de paramètres.

Enfin, la figure 5.38 indique les courbes du temps d'exécution de Klast (courbe 1), Klast dans sa variante 'seuil sur la moyenne des sous-mots' (courbe 2) et GKlast (courbe 3) en fonction, cette fois, du quorum et pour les valeurs suivantes de paramètres :

- matrice : PAM250 normalisée (voir figure 3.7)
- longueur des sous-mots : $w = 3$
- seuil : $t = 210$
- longueur des modèles recherchés : $k = 5$
- nombre de chaînes : $N = 20$
- nombre de trous autorisés (pour GKlast) : $e_s = 1$ (1 trou au plus par sous-mot de longueur w)

Le quorum quant à lui varie entre 50 et 100 avec un pas de 5. La présence exacte du modèle au moins une fois dans les chaînes n'est pas exigée.

5.5 Résumé des différentes versions

Un résumé de toutes les définitions de similarité entre des mots et des algorithmes permettant de résoudre le problème général énoncé dans la section 5.1 est donné dans le tableau 5.14.

Modèle	Relation entre modèle et occurrences	Algorithme	Lemme	Complexité en temps
Σ^k	I_k	KMR Moivre	 # 5.4.1	$O(n.N.\log k)$
Σ^k	$distance_{EH} \leq e$	HMoivre	# 5.4.2	$O(n.N.k^{e+1}. \mid \Sigma \mid^{e+1})$
Σ^k	$distance_{EL} \leq e$	LeMoivre	# 5.4.3 et 5.4.4	$O(n.N.(e+2).(2e+1).k^{e+1}. \mid \Sigma \mid^{e+1})$
Σ^k	$score_{\mathcal{M}} \geq t$	Klast	# 5.4.7	$O(n.N. \mid \Sigma \mid^w +$ $n.N. \mid \Sigma \mid .N_{Modi}(k))$
Σ^k	$score_{\mathcal{M}} \geq t$ $p_s \leq e_s$ $p_t \leq e_t$	GKlast		$O(n.N. \mid \Sigma \mid^w +$ $n.N.(e_s+2).(2e_t+1). \mid \Sigma \mid .N_{Modi}(k))$
C^k	R_k $\in$	KMRC et KMRS Poivre	 # 5.4.3 et 5.4.4	$O(n.N.\log k.g^{2k}.k)$ (KMRC) $O(n.N.k.g^k)$ (Poivre)
C^k	$distance_{EH} \leq e$	HPoivre	# 5.4.3 et 5.4.4	$O(n.N.k^{e+1}.p^{e+1}.g^k)$
C^k	$distance_{EL} \leq e$	LePoivre Modification 3	# 5.4.3 et 5.4.4	$O(n.N.(e+2).(2e+1).k^{e+1}.p^{e+1}.g^k)$ $O(n.N.(e+1).(e+2).k^{e+1}.p^{e+1}.g^k)$
CCP^k	$\in$	Combi	# 5.4.5	$O(n.N.k'_{max}.\log k'_{max}.2^{k'_{max}}+$ $m.k'_{max}.\log k'_{max}.p)$ ou $O(n.N.k.2^k + m.k.p)$
CCP^k	$distance_{EL} \leq e$	LeCombi	# 5.4.6	$O(n.N.(e+1).(e+2).k^{e+1}.p^{e+1}.2^k)$ $O(n.N.(e+1).(e+2).k^{e+1}.p^{e+1}.g^k)$

Tableau 5.14: Résumé de toutes les définitions de similarité et des algorithmes présentés ou rappelés dans ce chapitre (KMR/KMRC et KMRS).

Chapitre 6

Illustrations

First cicada :
life is
cruel, cruel, cruel.
Issa.

Come see
real flowers
of this painful world.
Basho.

The Penguin Book of Zen Poetry.
Lucien Stryk, Takashi Ikemoto (Eds. et trads.). Penguin, 1981.

Ce chapitre est consacré à l'illustration sur des cas concrets des divers algorithmes qui ont été décrits dans le chapitre 5.

Notre but n'est pas de présenter des résultats nouveaux, mais plutôt de tenter de porter un regard critique, au travers d'exemples variés, sur la méthode d'analyse de chaînes macromoléculaires par la recherche de mots communs similaires que nous avons adoptée, sur les définitions de ressemblance que nous avons établies et, enfin, sur le choix que nous avons fait d'explorer de manière combinatoire l'espace de ces chaînes.

Ces exemples concernent les deux grandes familles de macromolécules qui nous intéressent dans le cadre de ce travail : les acides nucléiques (ADN/ARN) et les protéines. D'une manière générale, nous connaissions *a priori* ce qu'il fallait 'trouver'. Certains exemples sont 'faciles' dans la mesure où ce qu'il y avait (de connu) à repérer était relativement aisé à localiser. Nous les présentons car ils permettent d'illustrer, soit certains points de notre approche, soit certaines difficultés liées à la nature des objets que nous devons analyser ou aux opérations situées en amont ou en aval de la comparaison proprement dite (établissement d'un ensemble de données, présentation et interprétation des résultats). D'autres exemples ont été choisis parce qu'ils sont considérés comme 'difficiles', ce qui nous a permis de tester nos méthodes dans des conditions plus extrêmes et, par conséquent, de mieux en mesurer leurs limites actuelles. Ces limites sont parfois techniques mais elles peuvent également être biologiques. Nous discutons les deux aspects.

La présentation adoptée suit l'ordre des exemples et non celui des méthodes. Une première partie est ainsi consacrée à l'ADN et à l'ARN et examine divers cas rencontrés dans la littérature ou suggérés par des biologistes proches. Des exemples concernant les protéines sont traités dans une seconde partie. Chacun de ces ensembles d'exemples est soumis à un ou plusieurs

des algorithmes que nous avons développés et, lorsque cela est possible, à ceux élaborés par d'autres. Chaque analyse est précédée d'une courte présentation de la famille de macromolécules examinée et du 'problème' biologique. Les résultats obtenus, ainsi que le champ d'application et les limitations des divers algorithmes employés, sont ensuite examinés. Cette analyse nous fournit parfois une occasion d'affiner la description sommaire donnée au début de ce travail des mécanismes biologiques mis en jeu et a pour objectif de mieux en faire sentir la complexité ainsi que la nature des problèmes algorithmiques que cette complexité soulève.

Par deux fois nous avons ressenti le besoin d'étendre de manière importante les algorithmes décrits dans le chapitre précédent. Nous avons reporté la présentation de ces extensions à ce chapitre-ci car nous voulions ainsi illustrer le fait que, même lorsque nous décidons de travailler avec des algorithmes combinatoires exacts et exhaustifs dans un certain sens, c'est la biologie qui doit préciser ce sens et influencer les définitions de ressemblance. Nous souhaitions également montrer la nécessité d'aller plus loin dans ce qui a été fait. L'une de ces extensions concerne exclusivement les chaînes nucléiques et est donc présentée en introduction aux exemples traitant de l'ADN et de l'ARN. L'autre extension pourrait s'appliquer aux deux types de chaînes macromoléculaires, mais n'a été utilisée pour l'instant que dans le cadre d'une étude sur des protéines. Elle est donc introduite au début de la section consacrée à ces molécules.

Observons enfin que l'approche est ici très naïve, ce n'est pas celle d'un biologiste. En particulier, les analyses que nous faisons servent les propos indiqués ci-dessus et uniquement ceux-ci. Du point de vue de la biologie, elles ne sont en aucun cas fines ou complètes.

6.1 Une observation à propos de la notation : un retour au vocabulaire de la biologie

Avant de présenter les illustrations de nos méthodes, il convient d'établir quelques points concernant le vocabulaire que nous allons employer dans ce chapitre. Jusqu'à présent, nous avons utilisé le terme de chaîne pour dénoter un certain codage des chaînes nucléotidiques ou polypeptidiques. La raison de ce choix était que ce codage pouvait représenter la structure, primaire aussi bien que tertiaire, des macromolécules. Nous ne pouvions ainsi nous servir du terme de séquence plus couramment employé en biologie puisque celui-ci est exclusivement associé à la structure primaire. Cependant dans les illustrations qui vont suivre, c'est toujours avec la structure primaire que nous allons travailler, aussi nous avons décidé de revenir à l'expression de séquence qui semble plus naturelle dans un contexte biologique que celle d'une chaîne. Pour la même raison, nous emploierons plus volontiers le terme de motif pour désigner un mot, surtout lorsque celui-ci sera associé à un signal biologique. Lorsque ce motif fera référence à la structure secondaire ou tertiaire plutôt qu'à la séquence, nous préciserons qu'il s'agit d'un motif structural. Enfin, lorsqu'aucune ambiguité n'est possible, nous utiliserons le terme de séquence pour désigner un motif considéré comme tel. Par exemple, nous parlerons de la séquence impliquée dans la régulation d'un gène même si cette séquence représente en fait un motif dans une séquence plus longue.

6.2 ADN et ARN

6.2.1 Introduction et nouvelle extension algorithmique

6.2.1.1 Le problème général

Lorsque nous travaillons avec des séquences d'ADN ou d'ARN, nous recherchons très fréquemment à identifier des signaux, par exemple de promotion de la transcription ou de la traduction. En général, nous essayons de localiser des mots qui apparaissent plus conservés dans un ensemble de séquences dont nous pouvons supposer qu'elles possèdent la même activité biologique. Repérer des signaux, les caractériser, n'est bien sûr pas la seule problématique associée à ce type de séquences, mais c'est une des plus importantes et c'est essentiellement celle qui nous concerne ici.

Le plus souvent, pour un lot de séquences à comparer, nous devons localiser un signal, ou un ensemble relativement restreint de signaux situés les uns à la suite des autres le long des séquences. Une approche naturelle du problème consiste alors, à partir d'une certaine définition de ressemblance entre mots, à rechercher tous les plus longs mots qui sont similaires suivant cette définition. Dans notre cas, cette démarche se ramène, ainsi que nous l'avons vu, à une recherche des plus longs modèles présents dans au moins un certain nombre de ces séquences.

Dans cette optique, les séquences d'ADN présentent un problème supplémentaire lié à leur structure en double hélice composée de deux brins anti-parallèles, et au fait que l'on ne connaît pas nécessairement lequel de ces deux brins il faut 'lire'. Les séquences présentes dans les banques ne correspondent pas toujours au brin 'actif', c'est-à-dire, porteur de l'information recherchée. De plus, dans certains mécanismes, les deux brins peuvent se trouver impliqués simultanément. Nous en verrons un exemple avec les promoteurs de la transcription.

Cela signifie que, dans nos algorithmes traitant de séquences d'ADN, nous devons pouvoir 'lire' les séquences que nous comparons dans les deux sens (c'est-à-dire, lire la séquence donnée au départ, et lire son complémentaire inversé qui correspond à la séquence du brin opposé de l'hélice). En termes de modèles et de leurs occurrences, ceci signifie qu'un mot va désormais être une occurrence d'un modèle soit parce que, comme avant, il est lui-même une occurrence dans le sens d'une des définitions de ressemblance que nous avons établies, soit parce que son complémentaire inversé l'est. Par exemple, si nos modèles sont des mots (notés m), et que nous avons $m = AAGTCT$, alors les mots $AAGTCT$ et $AGACTT$ sont tous deux des occurrences de m, le premier de manière exacte et le second en tant que complémentaire inversé. Nous esquissons donc dans la section suivante l'extension à apporter à nos algorithmes afin qu'ils puissent effectuer cette 'lecture dans les deux sens'.

6.2.1.2 Esquisse de la nouvelle extension algorithmique

Cette extension s'applique à ceux parmi nos algorithmes qui sont habituellement employés pour comparer des séquences nucléiques. Elle concerne donc H/LeMoivre (section 5.4.2.1), Combi (section 5.4.2.2) et LeCombi (section 5.4.2.3). Nous notons l'un quelconque de ces algorithmes par X.

Une façon simple de procéder serait bien sûr de faire tourner l'algorithme X deux fois, une fois sur les chaînes données au départ, une seconde fois sur ces mêmes chaînes inversées et complémentées, puis de vérifier le quorum sur l'union des deux ensembles en ne comptant qu'une fois la présence d'un modèle en même temps sur une chaîne et sur son complémentaire inversé.

Cette approche n'est cependant pas appropriée lorsque nous voulons obtenir les plus longs modèles et il est donc plus intéressant d'étendre l'algorithme de telle sorte que la double lecture

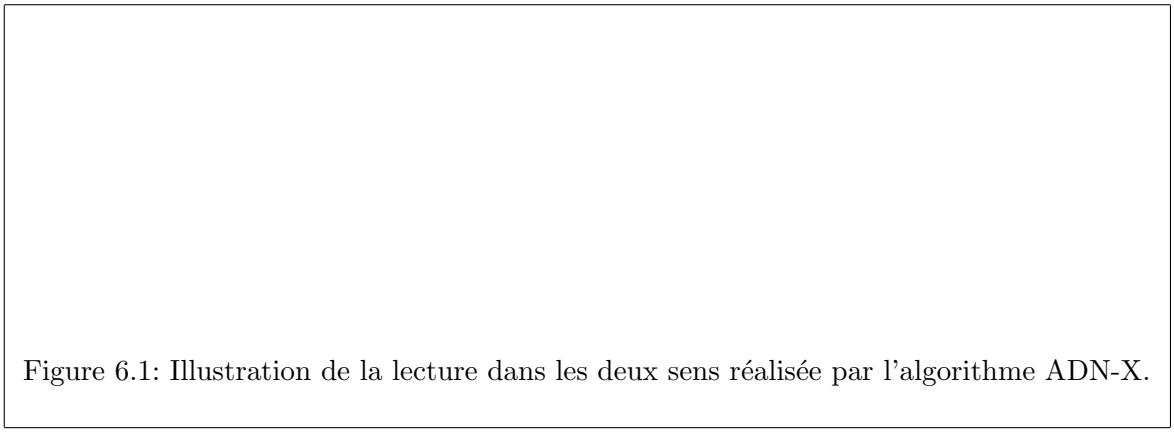


Figure 6.1: Illustration de la lecture dans les deux sens réalisée par l'algorithme ADN-X.

se fasse de manière simultanée lors de l'établissement de chaque modèle. Cette façon de faire présente un avantage supplémentaire : elle nous permet d'envisager une modification beaucoup plus importante de ces algorithmes permettant de localiser des structures symétriques miroir avec les mêmes définitions de ressemblance utilisées dans ce travail. Ces symétries peuvent représenter, soit des palindromes biologiques, soit faire partie de régions composées d'une suite contiguë de répétitions d'un même motif court, direct ou inversé (ces régions sont appelées des micro-satellites [Duret, 1995]).

L'extension des algorithmes de manière directe consiste alors à attacher une variable booléenne D ('Drapeau') à chaque occurrence d'un modèle M, et à procéder à la lecture du prochain symbole sur une séquence, soit dans le sens direct, soit dans le sens inverse (dans ce cas, ce sont les bases complémentaires de celles 'vues' qui sont effectivement 'lues'), suivant la valeur de D. Une illustration de cette double lecture est donnée dans la figure 6.1. La vérification du quorum se réalise ensuite de manière similaire aux versions originales. Nous notons ces extensions ADN-X. Leur complexité théorique est la même que celle des versions originales. Dans le pire des cas, elle est multipliée par une constante égale à 2.

6.2.2 Les promoteurs de *Bacillus subtilis*

6.2.2.1 Introduction

Nous avons vu dans le chapitre 2 que l'expression des gènes correspond à un transfert de l'information génétique de l'ADN aux ARN, puis des ARN aux protéines. Les molécules d'ARN sont synthétisées en utilisant la séquence nucléotidique de l'un des brins de l'ADN comme matrice dans une réaction de polymérisation réalisée par des enzymes appelées ARN polymérases. Le processus par lequel les molécules d'ARN sont initiées, allongées et terminées est appelé la transcription.

La première étape de cette transcription consiste en la fixation de l'ARN polymérase sur la molécule d'ADN. Cette fixation est réalisée au niveau de sites particuliers appelés promoteurs constitués de séquences spécifiques qui correspondent à des parties de la macromolécule d'ADN qui interagit avec l'enzyme. Un site en particulier connu pour être présent chez tous les organismes se situe à une distance de 5 à 10 bases en amont de la première base transcrite en ARN. Il est appelé la boîte de Pribnow pour les procaryotes ou la boîte de Hogness pour les eucaryotes (on parle aussi plus simplement de la séquence -10). Toutes les séquences identifiées à ce jour correspondant à cette boîte représentent des variations plus ou moins floues du motif

TATAAT ou *TATAAAT* respectivement. Cette boîte est supposée être une séquence permettant d'orienter l'ARN polymérase. C'est également la région où la double hélice de l'ADN s'ouvre pour former le complexe enzyme-ADN ouvert et autoriser ainsi le processus de transcription.

Une seconde séquence importante, présente dans la plupart des promoteurs de procaryotes, est appelée la séquence -35 parce qu'elle est localisée à environ 35 bases en amont de la première base copiée en ARN. La fonction précise de ce site n'est pas claire mais il est probable qu'une sous-unité de l'enzyme se fixe en premier lieu au niveau de la séquence -35 permettant ensuite à la région appropriée de la polymérase d'entrer en contact avec la boîte de Pribnow. Une fois fixée sur cette boîte, l'ARN polymérase se dissocie probablement du site de reconnaissance précédent.

Lorsque le promoteur est dépourvu de la séquence -35, un effecteur activateur (voir la section 2.1.1.3) est nécessaire à l'initiation de la transcription. Bien que le mécanisme d'action de cet effecteur ne soit pas encore tout à fait compris, il est probable que celui-ci se fixe en premier sur une autre séquence spécifique du promoteur puisque c'est cette interaction qui doit faciliter celle de la polymérase avec la boîte de Pribnow. La protéine CRP constitue un exemple d'une telle protéine effectrice et nous parlerons de son site de fixation un peu plus loin dans le chapitre. La protéine CRP est également impliquée dans la régulation de nombreux gènes intervenant dans le métabolisme des sucres, en l'absence ou présence de la séquence -35, en tant qu'activateur ou en tant que répresseur.

Outre les séquences situées à -10 et -35 du début de la transcription et celles où viennent parfois se fixer des effecteurs, il est possible que d'autres caractéristiques de l'ADN proches de la région devant être transcrite affectent également cette transcription. Galas *et al.* discutent ainsi [Galas *et al.*, 1985] l'existence d'une séquence qui interviendrait dans la promotion du processus de transcription chez *Escherichia coli* et qui se localiserait à -44, c'est-à-dire environ 9 bases en amont de la séquence -35.

Dans l'exemple discuté ici, nous avons d'abord voulu tester si nos algorithmes étaient capables de repérer la présence des séquences 'classiques' (celles localisées à -10 et -35) des promoteurs de *Bacillus subtilis* qui sont, en général, bien conservées. Nous avons ensuite voulu essayer de déterminer si d'autres séquences ne se trouvaient pas également anormalement conservées en amont ou en aval de ces sites et pourraient ainsi être impliquées dans la promotion de la transcription.

L'ensemble avec lequel nous avons travaillé est composé d'un lot de 142 séquences de longueur moyenne 120 nucléotides correspondant à 100 bases en amont de la première base transcrite et de 20 bases en aval. Ces séquences ont été répertoriées par Helmann [Helmann, 1995] et sont disponibles par ftp (URL http://www.bio.cornell.edu/microbio/helmann/helmann.htlm). Helmann présente en fait 246 séquences dont 94 sont hypothétiques ('putative promoters'). Nous avons choisi de travailler uniquement avec les 142 dont la fonction de promotion a pu être validée expérimentalement. En réalité, nous avons réduit cet ensemble aux 131 séquences du lot ayant une longueur supérieure à 100 bases (celles éliminées ne comportent en moyenne qu'une cinquantaine de bases et ne présentent que les sites à -10 et -35). Enfin, nous n'avons conservé sur chaque séquence que les 100 bases en amont de la première base transcrite.

Sur ces séquences, Helmann a réalisé un travail de comptage des divers nucléotides présents à chaque position des séquences après alignement de celles-ci sur la première base transcrite. Cela lui a permis de mettre en évidence :

- les sites -10 et -35 (très exactement -12 et -35 et dont les consensus sont respectivement *TAtAAT* et *TTGaca* — les bases en majuscules sont présentes dans plus de 70% des promoteurs);

- la présence d'un dinucléotide, TG, assez conservé à -15;
- l'existence de plusieurs régions riches en A en amont du site -35 (situées autour des positions -43, -54 et -65) ainsi que des régions riches en T localisées entre celles riches en A.

Sa méthode d'analyse ne lui donne toutefois pas les moyens de vérifier si une structure plus complexe existe à cet endroit.

Nous résumons dans la prochaine section les premiers résultats obtenus.

6.2.2.2 Résultats

Notre première approche lorsque nous avons voulu tester nos algorithmes sur ce jeu d'essai a consisté tout naturellement à rechercher les modèles présents dans au moins un certain nombre de séquences. À l'époque nous développions Combi (section 5.4.2.2); c'est donc avec des modèles définis sur une couverture combinatoire pondérée que nous avons initialement travaillé. Nous avions en fait établi un certain nombre de couvertures pondérées dont un exemple est la couverture suivante :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{\Sigma\}, 0), \\ (S, 1)\ \textit{pour tous les}\ (10)\ \textit{autres ensembles} \end{array} \right\}.$$

De plus, nous avions travaillé avec la variante de Combi décrite dans la section 5.4.2.2.6.1, ce qui veut dire que la pondération sur les ensembles était limitée par la pondération sur le joker lors de la première étape de l'algorithme, ceci afin d'éviter d'obtenir des modèles trop dégénérés. Cette pondération initiale sur le joker était égale à 4, et donc le nombre total d'ensembles 'spéciaux' permis dans les modèles (les non singletons) était au plus de 4.

Nous avions alors recherché les modèles les plus longs présents dans au moins 70% des séquences d'un sous-ensemble des 142 composé de 10, 50, 100 et 140 séquences [Sagot and Viari,1996]. Ainsi que nous l'avons indiqué dans l'article et dans la section 5.4.2.2.7, le nombre de modèles trouvés lorsque $k = k_{max}$ varie considérablement suivant le nombre de séquences faisant partie du lot test. Nous avons argumenté que cette instabilité des résultats obtenus par rapport aux paramètres choisis, en particulier l'explosion du nombre de modèles dans certaines circonstances, n'est pas nécessairement une mauvaise chose et peut apporter une information importante. Cela est en particulier le cas lorsque, comme ici, nous bénéficions du fait que les séquences sont cadrées, c'est-à-dire, qu'elles peuvent être alignées de manière 'naturelle' sur la première base à transcrire.

Il est en effet possible, dans ce cas, d'établir un graphique indiquant la position où les modèles sont trouvés, ce qui nous permet, surtout lorsque le nombre de ces modèles est grand, d'avoir une idée des régions qui peuvent éventuellement être associées à un signal. Le graphique que avons obtenu pour un lot de 140 séquences, avec la couverture indiquée ci-dessus et q égal à 70%, est celui donné dans la figure 6.2. Cette courbe présente quatre pics nets, deux faisant référence au même signal situé à -10 (boîte de Pribnow) et -14 (dinucléotide conservé TG), un second pic à -33 et enfin un troisième autour de -46. Sachant d'après Helmann que la région autour de la position -45 est particulièrement riche en A et en T (plus que le reste de la séquence qui présente déjà un biais en composition en faveur de ces deux bases), la question que nous nous sommes posée alors était : est-ce que ce pic est dû au fort biais en composition des séquences dans cette région ou bien y a-t-il une structure, c'est-à-dire un vrai motif, à cet endroit?

Figure 6.2: Graphe du nombre des modèles (plus longs trouvés avec la couverture CCP donnée dans le texte, $k = k_{max}$, $q = 70\%$ et pour 140 séquences). La position 0 en abscisse indique l'endroit où débute la transcription.

Pour essayer d'y répondre, nous avons par la suite fait une recherche plus exhaustive des modèles présents dans ces séquences et pour cela nous avons adopté la stratégie suivante :

- utiliser non seulement Combi mais aussi HMoivre (section 5.4.2.1.3) et LeCombi (section 5.4.2.2) car une de ces approches pouvait être plus appropriée que les autres pour repérer les signaux présents dans ces régions;
- découper les séquences en trois fragments de longueur 15, centrés sur les positions -10, -33 et -46 respectivement, constituant ainsi trois nouveaux lots de séquences;
- rechercher dans chacun des trois lots tous les modèles de longueur supérieure à une certaine valeur (entre 4 et 6 suivant la méthode);
- enfin, essayer de déterminer dans quelle mesure les modèles trouvés sont dus au hasard, cela en effectuant la même recherche sur des séquences aléatoires de même longueur et de même composition en mono et en dinucléotides que les fragments originaux (cent de ces séquences aléatoires sont produites pour chaque fragment de l'ensemble de départ).

Cette dernière évaluation a été réalisée en comparant des tables de contingence qui permettent de déterminer si une première variable (ici la présence d'un modèle dans un ensemble de séquences) dépend d'une seconde variable (ici le fait que ces séquences sont de 'vraies' séquences où un certain nombre de signaux se trouvent présents). Si cela est le cas, nous pouvons supposer que la présence du modèle dans les séquences n'est pas dû au hasard mais peut être liée au signal recherché [Clarke and Cooke, 1992]. Pour chaque modèle M, nous disposons alors de deux tables de 2×2 cellules chaque. La première, notée $\mathcal{O}$ (observation), indique le nombre de séquences où le modèle M est présent au moins une fois (resp. absent) dans l'ensemble des 'vraies' séquences, et le nombre de fois où il est présent (resp. absent) dans l'ensemble des 'fausses' séquences, c'est-à-dire, des séquences aléatoires. La seconde table, notée $\mathcal{A}$ (attendu), correspond à l'hypothèse nulle, c'est-à-dire à l'hypothèse d'indépendance des deux variables indiquées ci-dessus. Les cellules de cette table valent $\frac{n_i.n_j}{N}$ où n_i indique le nombre total de séquences où M est présent (absent), n_j le nombre de séquences 'vraies' ('fausses') et N le nombre total de séquences (voir figure 6.3). Ces deux tables sont ensuite comparées en effectuant un test du χ^2 (à 1 degré de liberté) :

$$\chi^2 = \sum_{\substack{toutes\ les \\ cellules\ i,j}} \frac{(\mathcal{O}_{ij} - \mathcal{A}_{ij})^2}{\mathcal{A}_{ij}}.$$

Les résultats obtenus sont approximativement les mêmes que l'on travaille avec HMoivre ou avec Combi. Dans les deux cas, une interprétation vraiment claire n'est possible que lorsque relativement peu de souplesse est autorisée. Le nombre d'erreurs permis dans HMoivre (substitutions uniquement car il n'y a, en principe, pas de trous à l'intérieur des sites de promotion) est donc égal à 0 ou 1 et les couvertures combinatoires adoptées dans le cas de Combi correspondent à la variante de la section 5.4.2.2.6.2 (pondération par cardinalité) et sont très faiblement dégénérées (au maximum 2 sous-ensembles autres que les singletons sont permis dans un modèle). Ces résultats sont les suivants :

- les modèles trouvés dans chaque cas sont de longueur relativement petite (6 ou 7 bases) même pour des quorums assez bas (5-20%);
- les modèles trouvés dans les régions situées autour des pics à -10 et -35 sont significatifs par rapport à l'hypothèse nulle (une indication de quelques-uns de leurs noms, nombre

Table $\mathcal{O}$:

	Présents	Absents	
Vraies	P_V	A_V	n_V
Fausses	P_F	A_F	n_F
	n_P	n_A	

$$n_V + n_F = n_P + n_A = N$$

Table $\mathcal{A}$:

	Présents	Absents	
Vraies	$\frac{n_V.n_P}{N}$	$\frac{n_V.n_A}{N}$	n_V
Fausses	$\frac{n_F.n_P}{N}$	$\frac{n_F.n_A}{N}$	n_F
	n_P	n_A	

Figure 6.3: Tables de contingence pour évaluer les modèles de promoteurs obtenus avec nos algorithmes.

de 'vraies' et de 'fausses' séquences où ils sont présents et probabilité par rapport à l'hypothèse nulle (i.e. probabilité d'obtenir une valeur de χ^2 supérieure à celle observée) pour HMoivre avec 0 erreur, q = 5% et un mélange en dinucléotides est donnée dans le tableau 6.1);

- les modèles trouvés dans la région du pic à -45 sont présents dans un nombre très faible de séquences (environ 5%) et ne sont pas significatifs lorsqu'évalués par rapport à des séquences mélangées en préservant la composition en dinucléotides.

Ces résultats semblent donc indiquer que, du moins en ce qui concerne les petits motifs, la région autour de -45 ne présente aucune structure qui ne pourrait être expliquée par le simple biais de la composition des séquences à cet endroit.

Les observations réalisées avec LeCombi sont, par contre, très différentes, surtout lorsqu'un nombre relativement grand de jokers est autorisé. La couverture utilisée est maintenant la suivante :

$$CCPR = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{A, C, G, T\}, J), \\ (S, 0)\ \textit{pour tous les autres ensembles} \end{array} \right\}$$

avec J entre 4 et 20. Les modèles trouvés, même lorsqu'aucune erreur n'est autorisée, sont plus longs (11 pour 4 jokers et $q = 20\%$, 27 pour 20 jokers et $q = 20\%$) et une nette structure apparaît alors avant le site -35. La démarche que nous avons adoptée pour essayer de la caractériser a consisté à :

- établir deux lots de 131 séquences, le premier (lot A) est l'original (longueur des séquences égale à 100 bases) et présente les 3 pics, le second (lot B) représente les mêmes séquences mais chacune ne comporte plus que 30 bases entre -35 et -65 (ce second lot ne présente donc que le pic à -45 et correspond à la région riche en A et T);

Modèle (site -10)	Prés. séq. vraies	Prés. séq. fausses	Probabilité χ^2
$TATAAT$	23	3,6	10^{-15}
$CTATAAT$	9	0,8	10^{-9}
$ATAAT$	50	21	10^{-9}
$TATAATA$	17	3,3	10^{-9}
$TATAATG$	10	1,6	10^{-7}
Modèle (site -35)	Prés. séq. vraies	Prés. séq. fausses	Probabilité χ^2
$TTGACA$	21	1,6	10^{-23}
$TTGAC$	39	10,2	10^{-15}
$GACA$	31	6,9	10^{-14}
$GTTGAC$	15	2,9	10^{-10}
$ATTGAC$	13	1,8	10^{-10}

Tableau 6.1: Quelques modèles trouvés relatifs aux sites de promoteurs -10 et -35. L'algorithme utilisé ici est HMoivre avec 0 erreur et un quorum de 5%. Les modèles devaient avoir une longueur supérieure ou égale à 4. Le nombre d'occurrences fausses est une moyenne sur cent mélanges.

- rechercher ensuite les plus longs modèles présents dans au moins 20% des séquences du lot A ou B avec 0 erreur et 4, 8 ou 20 jokers;
- effectuer un classement des modèles trouvés basé sur l'écart observé entre le nombre d'occurrences sur l'ensemble des 'vraies' séquences et celui sur l'ensemble des séquences aléatoires qui préservent la composition en dinucléotides, cet écart étant mesuré par un χ^2;
- enfin, dans certains cas, 'lire' les séquences dans les deux sens : dans le sens direct et dans le sens complémentaire inversé.

La justification pour ce dernier point dans le cas des promoteurs est la suivante. Nous avons vu auparavant que le site -10 est la séquence d'orientation de la polymérase, et il est probable que l'orientation du site -35 soit également importante. Il est très possible cependant que cela ne soit plus le cas pour un signal qui serait présent autour de la position à -45, et que la protéine qui viendrait éventuellement s'y fixer pourrait tout aussi bien se mettre en contact avec le motif correspondant sur l'autre brin de la double hélice d'ADN. C'est ce qui semble être le cas pour certains promoteurs de *coli* ainsi que le suggère Galas [Galas *et al.*, 1985]. Il est donc nécessaire de 'lire' la séquence dans les deux sens, sachant que cela correspond à une lecture simultanée des deux brins. Notons que, dans le cas de cet exemple, cette double lecture ne change que peu les résultats pour les petits modèles observés dans la région -45 qui demeurent non significatifs. Ceux repérés par LeCombi par contre sont plus sensiblement affectés, surtout pour les motifs moins longs obtenus en autorisant 4 jokers : certains représentent des occurrences du modèle sur le brin opposé de l'ADN.

Nous avons choisi de présenter sous forme de graphiques les résultats de quelques modèles

sélectionnés. Considérons un de ces modèles : $TTGN_{20}TATA$. Nous pouvons nous poser plusieurs questions à son propos :

- lorsqu'un symbole (un singleton) apparaît dans ce modèle, par exemple le premier T, pouvons-nous affirmer que ce symbole est réellement important, c'est-à-dire, qu'il ne peut être substitué par un autre? Pour essayer de le déterminer, l'idée est ici de remplacer un à un, de manière indépendante, chacun des symboles non joker du modèle par les trois autres symboles possibles et de compter le nombre d'occurrences du nouveau modèle ainsi constitué. Cette opération produit les graphiques notés 'a'.

- lorsqu'un modèle présente un joker à une certaine position, en quoi cette position est-elle vraiment non spécifique? Cette fois l'idée est de calculer la matrice des fréquences observées de chaque symbole en chaque position des occurrences du modèle lorsqu'aucune substitution n'est autorisée. D'une manière plus générale, nous pouvons également calculer cette matrice pour les occurrences identifiées lorsque $e > 0$ substitutions sont permises. Dans les graphiques notés 'b' donnés ci-après, nous avons choisi $e = 1$.

- enfin, lorsque plusieurs jokers consécutifs apparaissent dans un modèle (par exemple N_{20} dans $TTGN_{20}TATA$), est-ce que ce nombre de jokers contigus est précisément fixé, ou bien peut-il varier dans un certain intervalle? En effet, une suite de jokers contigus peut être considérée comme un 'espace' (en anglais 'spacer') entre deux motifs plus stricts, et nous pouvons légitimement nous interroger sur l'optimalité de la longueur de cet 'espace'. L'idée afin de nous assurer de cette optimalité est de rechercher le modèle en faisant varier la longueur de l"espace'. La quantitée mesurée pour chaque valeur de cette longueur est alors le χ^2 entre le nombre d'occurrences observées, d'une part sur les 'vraies' séquences, d'autre part sur les séquences aléatoires. Ceci définit les graphiques notés 'c' donnés ci-après.

En résumé, nous avons établi pour chaque modèle sélectionné trois types de graphiques :

Graphique a : le nombre d'occurrences observées (lot A ou B) lorsque une (une seulement) des bases (pas le joker) est mutée en une autre base à tour de rôle;

Graphique b : fréquences des bases observées en chaque position dans les occurrences du modèle (lot A ou B) avec cette fois 1 substitution autorisée;

Graphiques c : χ^2 entre le nombre d'occurrences observées (lot A ou B), d'une part sur les séquences 'vraies' et d'autre part sur les séquences aléatoires, lorsque la longueur l des suites contiguës d'au moins 2 jokers (une suite seulement à chaque fois) est variée avec un pas de 1 entre deux valeurs égales à $min\{l - 10, 1\}$ et $l + 10$.

Le tableau 6.2 résume les noms, le nombre d'occurrences et le χ^2 des modèles de longueur maximale parmi les plus intéressants identifiés par LeCombi. Les graphiques (de types 'a', 'b' ou 'c') associés à chacun d'eux sont donnés dans les pages suivantes.

Par exemple, la figure 6.4 représente les graphiques relatifs au modèle $TTGN_{20}TATA$, qui est un des modèles les plus significatifs trouvés dans le lot A avec $q = 20\%$ et en autorisant 20 jokers mais pas d'erreurs (5798 secondes sur une Sparc Station 20). Ce modèle correspond aux trois premières bases du site -35 suivies d'une suite de jokers (notés N) puis des quatre premières bases du site -10. Il est exactement présent dans 28 des 131 séquences, et nous pouvons observer que les bases associées aux jokers sont plus fréquemment des A ou des T (graphique b) sauf aux positions 5 et 23 du modèle où les bases C et G respectivement sont légèrement

Modèle	Occ. sur vraies	Occ. sur fausses	χ^2
$TTGN_{20}TATA$	28	2.69	238.14
$TNTN_3AAAAN_{16}A$	33	8.46	71.18
$TTTTN_2ANAAN_5T$	16	1.24	175,69
$ANAN_3TNTN_3AAAA$	10	0.92	89,62
$AAANAN_7AAAA$	10	1.48	49,05
$TNTTTN_2AANA$	30	6.83	78,60
$TNTTN_2AAANA$	30	6.83	78,60
$TNTTN_3AAAA$	34	8.65	74,29

Tableau 6.2: Quelques modèles trouvés avec LeCombi pour $k = k_{max}$, $q = 20\%$ avec 20, 8 et 4 jokers autorisés.

plus souvent rencontrées. Ce G correspond en fait à celui du dinucléotide conservé TG à -14 (voir la figure 6.2). D'autre part, il est intéressant de noter que le nombre de jokers centraux (N_{20} dans le modèle) est rigoureusement conservé à 20. Cet écart de 20 bases représente en réalité un écart de 17 entre la fin du premier site et le début du second si l'on considère que le site -35 possède 6 bases, les trois dernières n'apparaissant pas dans ce modèle. Cet écart est ainsi pratiquement constant pour les promoteurs dont le site à -35 débute par TTG et celui à -10 par $TATA$. Helmann montre que cela demeure vrai en général et que cette distance de 17 bases entre les deux sites correspond à une valeur optimale chez *Bacillus subtilis* de la même manière que chez *Escherichia coli*.

Les figures suivantes (figures 6.5 à 6.11) correspondent toutes à des modèles repérés dans la région autour du pic observé à -46 (le lot utilisé est le A pour la figure 6.5 et le B dans tous les autres cas, la lecture est directe pour les quatre premières figures et dans les deux sens pour les trois dernières), pour un nombre de jokers égal à 20 (figure 6.5), 8 (figures 6.6, 6.7 et 6.8) et 4 (figures 6.9, 6.10 et 6.11) et toujours avec $q = 20\%$ et $e = 0$ (les temps d'exécution ont été de 1038 secondes pour 8 jokers et de 129 secondes pour 4). Ces figures permettent de dégager la présence d'une nette structure dans la région à -45, du moins en ce qui concerne une partie des promoteurs de *subtilis*. Cette structure peut être caractérisée par :

- un motif central M_C composé d'une suite de 4 à 5 T, suivie par 2 à 3 N puis par une suite de 4 à 5 A, c'est-à-dire $M_C = T_{4-5}N_{2-3}A_{4-5}$;
- deux motifs latéraux, chacun d'eux englobant le motif central mais apparemment ne pouvant apparaître tous deux simultanément :
  - $A_{4-5}N_3M_C$;
  - $M_CN_{3-6}T_{2-3}$;
- un motif composé d'une suite de A, suivie d'une suite de jokers puis de la seconde moitié du motif central : $A_{4-5}N_{6-8}A_{4-5}$.

Figure 6.4: Graphiques correspondant au modèle $TTGN_{20}TATA$ (χ^2 = 238,14) identifié dans le lot A par LeCombi (lecture directe uniquement) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 20, q = 20% et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

Notons également que les 2 ou 3 N situés entre les T et les A représentent le plus souvent un T suivi par un A, avec éventuellement une position au centre pouvant être indifféremment un T ou un A. Enfin, le pic à -45 correspond généralement au motif central; plus précisément, c'est le premier symbole de la suite A_{4-5} qui se trouve en général à cet endroit.

Cette étude demeure bien sûr tout à fait préliminaire et une analyse plus précise serait nécessaire pour confirmer ces premiers résultats. Il pourrait être éventuellement intéressant de classer les gènes sur la base des différents motifs observés autour du pic à -45.

6.2.2.3 Discussion

Ce premier exemple nous conduit à faire un certain nombre d'observations concernant nos algorithmes.

La première est que nous avons été amenés, au cours de cette analyse, à tester plusieurs de ces algorithmes : certains (HMoivre, Combi) se sont révélés plus intéressants pour repérer les petits motifs plus conservés correspondant aux sites -10 et -35, alors que c'est LeCombi (avec des jokers) qui a permis de dégager une structure dans la région à -45. Le site à cet endroit, s'il se confirme qu'il existe, semble être plus long et, surtout, il présente des positions apparemment non-spécifiques. Ces positions forment souvent des suites contiguës qui peuvent ne pas être de taille constante (bien que celles des modèles que nous repérons sont en général quasi-optimales — voir les graphiques c des figures 6.4 à 6.11 précédentes). Il serait alors intéressant de disposer d'un algorithme qui permette de traiter de telles suites, c'est-à-dire, qui permette de traiter le problème des jokers de longueur variable (en anglais 'variable length don't care symbols'). Nous pouvons à l'heure actuelle le faire de manière indirecte en autorisant des trous, mais le concept n'est pas tout à fait le même et il est essentiel de maintenir les deux idées séparémment. D'autre part, une telle approche serait importante pour l'identification des motifs à -10 et -35 puisque la promotion de la transcription est associée en réalité non à chacun de ces deux motifs isolément, mais à la structure générale : séquence -35, séquence -10 et espacement optimal (17 bases) entre les deux.

La seconde observation est que nous avons été obligés, dans tous les cas, de faire tourner plusieurs fois chacun des algorithmes en faisant varier les divers paramètres (quorum, nombre d'erreurs, couvertures et pondérations, nombre de jokers). Or le choix de ceux-ci demeure pour l'instant délicat. Est-ce que l'intuition et une certaine expérience des données sont les seuls guides dont nous puissions disposer? Est-ce que la seule stratégie possible consiste à lancer le maximum de tests, essayant de couvrir au mieux l'arsenal à notre disposition, pour ne conserver à la fin que les résultats qui paraissent les plus intéressants?

Quant à ces résultats eux-mêmes, ils se trouvent encore, d'une certaine façon, à l'état brut. Ils possèdent une richesse d'information qui n'est pas exploitée, un exemple plus précis de cela sera donné plus tard. Ils représentent également une quantité de données qui est parfois presque aussi considérable que celle dont nous disposions au départ, simplement mieux organisée, ou organisée différemment. Il faut donc souvent faire appel à d'autres techniques pour essayer de trier ce qui est 'vu' et de l'interpréter. Ces techniques (par exemple ici statistiques) sont en général automatisables, mais les liaisons entre les diverses étapes de l'analyse doivent encore souvent être faites à la main. Est-il possible de les automatiser, du moins en partie? En particulier, est-il possible de fournir une représentation des modèles repérés autre qu'une simple liste comme nous le faisons pour l'instant? Par exemple, nous avons fourni plusieurs modèles correspondant aux sites -10 et -35 et obtenus avec HMoivre qui visiblement sont des variants les uns par rapport aux autres. Pourrait-on les résumer en préservant toute l'information qu'ils contiennent? Pourrait-on faire de même avec les modèles fournis par Combi ou LeCombi?

Figure 6.5: Graphiques correspondant au modèle $TNTN_3AAAAN_{16}A$ (χ^2 = 71,18) identifié dans le lot A par LeCombi (lecture directe uniquement) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 20, $q = 20\%$ et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

Figure 6.6: Graphiques correspondant au modèle $TTTTN_2ANAAN_5T$ ($\chi^2 = 175{,}69$) identifié dans le lot B par LeCombi (lecture directe uniquement) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 8, $q = 20\%$ et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

Figure 6.7: Graphiques correspondant au modèle $ANAN_3TNTN_3AAAA$ (χ^2 = 89,62) identifié dans le lot B par LeCombi (lecture directe uniquement) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 8, q = 20% et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

Figure 6.8: Graphiques correspondant au modèle $AAANAN_7AAAA$ ($\chi^2 = 49{,}05$) identifié dans le lot B par LeCombi (lecture directe uniquement) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 8, $q = 20\%$ et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

Figure 6.9: Graphiques correspondant au modèle $TNTTTN_2AANA$ ($\chi^2 = 78{,}60$) identifié dans le lot B par ADN-LeCombi (lecture dans les deux sens) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 4, $q = 20\%$ et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

Figure 6.10: Graphiques correspondant au modèle $TNTTN_2AAANA$ (χ^2 = 78,60) identifié dans le lot B par ADN-LeCombi (lecture dans les deux sens) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 4, q = 20% et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

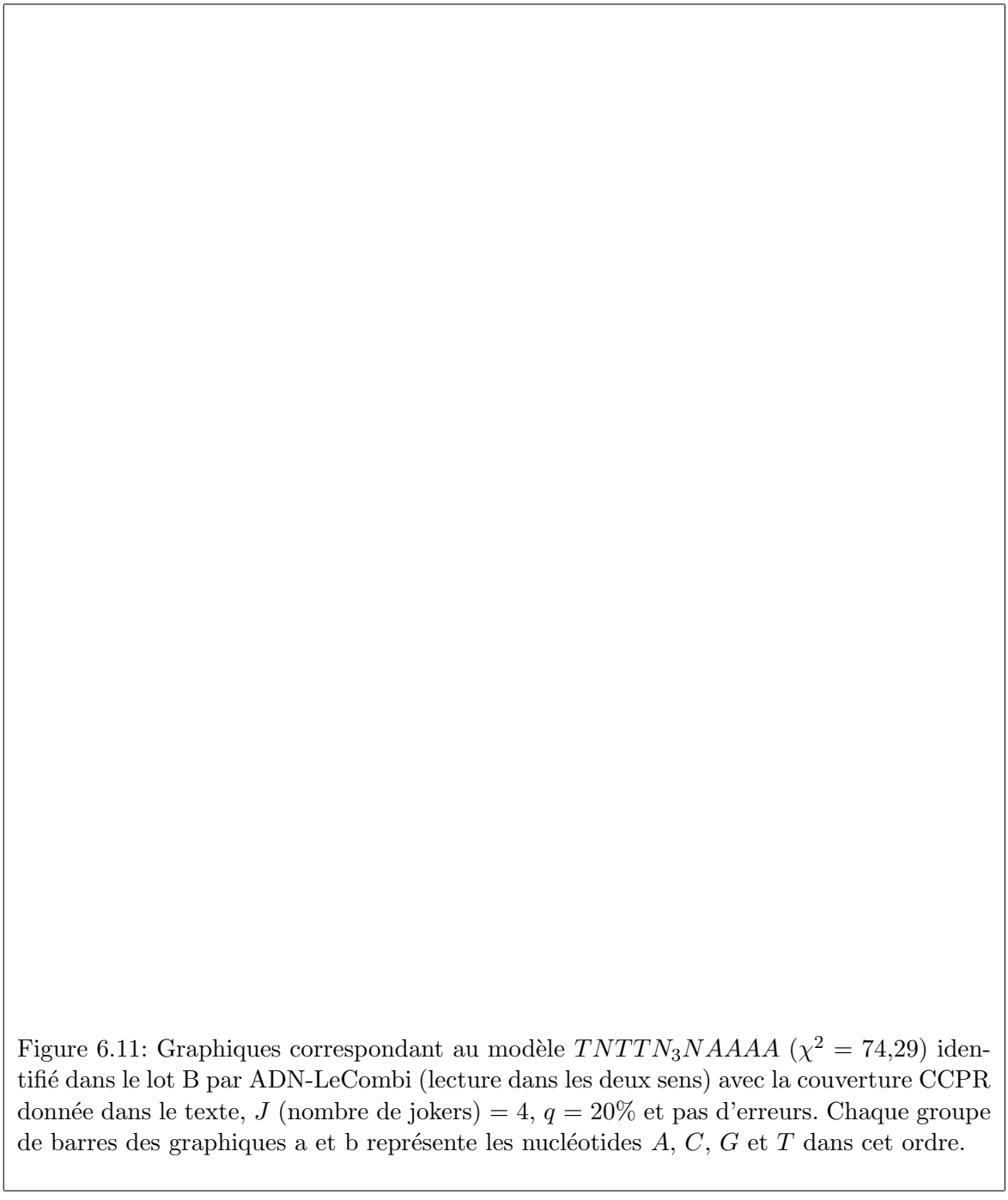


Figure 6.11: Graphiques correspondant au modèle $TNTTN_3NAAAA$ (χ^2 = 74,29) identifié dans le lot B par ADN-LeCombi (lecture dans les deux sens) avec la couverture CCPR donnée dans le texte, J (nombre de jokers) = 4, q = 20% et pas d'erreurs. Chaque groupe de barres des graphiques a et b représente les nucléotides A, C, G et T dans cet ordre.

Enfin, le bon grain se trouve souvent mélangé à l'ivraie dans ce qui est obtenu. Cela est particulièrement le cas lorsque nous devons autoriser beaucoup de souplesse dans la recherche de mots communs similaires. Ainsi le motif à -35, qui est bien plus 'faible' que celui à -10, n'a pu être repéré avec HMoivre et 0 erreur que parce que nous avons fait usage *a posteriori* de statistiques. Celles-ci permettent de le 'voir' parce qu'il comporte deux bases — une Cytosine et une Guanine — peu fréquentes dans toute la région des promoteurs. Le fait que les modèles correspondants aient moins d'occurrences est ainsi 'compensé' par le caractère 'anormal' de leur présence et permet de dire avec une certaine assurance qu'ils sont probablement intéressants. Si nous avions voulu utiliser HMoivre tout seul, nous aurions du autoriser au moins 2 substitutions et, surtout, fixer la longueur des modèles à une valeur relativement courte (autour de 6 ou 7 nucléotides) afin de localiser cette séquence -35 avec des quorums plus élevés. Dans ce cas cependant, et sur cet exemple, les signaux commencent alors à se mélanger et le bruit augmente. D'une façon générale, Combi et LeCombi semblent être plus discriminants dans la mesure où ils permettent plus de souplesse tout en fixant de manière plus forte certaines positions. Est-ce que cela signifie que la définition de ressemblance sur laquelle ils sont établis correspond mieux aux processus de reconnaissance biologique? C'est une question à laquelle nous ne savons répondre.

6.2.3 Les promoteurs dépendents d'une protéine d'activation catabolique de *Escherichia coli*

6.2.3.1 Introduction

Le nombre de protéines produites en une unité de temps par un organisme n'est pas constant. Il varie, pour un même gène, au cours du temps, et d'un gène à l'autre, de telle sorte que les besoins de la cellule soient satisfaits tout en évitant de réaliser des synthèses inutiles. Les mécanismes de régulation de l'expression des gènes peuvent être du type 'marche-arrêt' ou plus élaborés. Malgré leur diversité, il est possible de regrouper tous ces mécanismes en deux catégories : ceux dont la régulation est positive (activation) et ceux dont la régulation est négative (inhibition) (voir la section 2.1.1.3). Dans un système de régulation négatif, un inhibiteur présent dans la cellule bloque la transcription ou la traduction. Un antagoniste de l'inhibition, généralement appelé inducteur, est alors nécessaire pour initier un de ces deux processus. Dans un système de régulation positif, une molécule effectrice (qui peut être une protéine, une petite molécule ou un complexe moléculaire) active un promoteur. Les deux systèmes ne sont pas exclusifs et l'expression d'un gène peut être à la fois positivement et négativement régulée.

La bactérie *Escherichia coli* possède ainsi une protéine d'activation catabolique (CAP, en abbréviation de l'anglais 'Catabolic Activator Protein'), appelée également la protéine réceptrice de la protéine cyclique AMP (CRP, de l'anglais 'Cyclic-AMP Receptor Protein'), qui est impliquée dans la régulation de la transcription de quelques-uns de ses gènes. Ces derniers (isolés ou regroupés en une unité fonctionnelle appelée opéron) ne peuvent en effet être transcrits que si le complexe cAMP-CRP est fixé à l'ADN dans la région du promoteur (dans le cas des opérons, les gènes composant l'unité sont alors co-régulés et co-transcrits). En fait, ce complexe peut également agir comme un répresseur de la transcription de certains gènes, cela est le cas en particulier du gène qui réalise la propre synthèse des CRP [Botsford and Harman, 1992] [Combrugghe *et al.*, 1984], cependant cette répression est probablement due au fait que le site promoteur situé à -35 du gène inhibé chevauche celui où doit se fixer le complexe afin d'activer la transcription d'un autre gène situé à proximité du premier. Par ailleurs, deux sites ont parfois été identifiés pour un même gène, un site primaire et un site secondaire qui fixe la CRP moins fortement et qui peut participer à la localisation du site primaire par le complexe [Combrugghe *et al.*, 1984]. D'une façon plus générale, les promoteurs

CAP-dépendants peuvent être groupés en trois ou quatre classes [Combrugghe *et al.*, 1984] [Ebright, 1993] [Gaston *et al.*, 1990] suivant la localisation des sites fixateurs de la CRP, la fonction de celles-ci et leur mode d'action (le contact entre molécules pouvant intervenir uniquement entre la CRP et l'ADN, ou avoir lieu également entre la CRP et l'ARN polymérase, ou encore impliquer d'autres protéines qui viendraient se fixer à la suite les unes des autres sur l'ADN entre la CRP et la polymérase).

Ce qu'il est important de noter dans le cadre de notre analyse, c'est que tous ces sites présentent deux motifs principaux qui sont séparés par, en général, 6 bases. Le premier motif, $TGTGA$, est assez fortement conservé tandis que le second motif, moins conservé, est soit une version complémentaire inversée du second ($TCACA$), soit un autre type de séquence [Combrugghe *et al.*, 1984]. La possibilité d'une symétrie miroir entre les deux séquences composant le site s'explique par la nature dimérique de la protéine CRP qui interagit avec l'ADN sur 22 à 25 bases [Berg and von Hippel, 1988] [Combrugghe *et al.*, 1984]. Cette nature dimérique ne signifie cependant pas que les deux séquences nucléotidiques établissant le contact avec la CRP sont d'égales importances pour la fixation du complexe. La moindre conservation du second motif, résultant d'une asymétrie du site, peut ainsi jouer un rôle non négligeable dans cette fixation, en particulier lorsque qu'il y a également contact entre CRP et ARN polymérase [Berg and von Hippel, 1988] [Combrugghe *et al.*, 1984]. Dans ce cas, la fixation de la CRP à l'ADN du côté où viendra se placer la polymérase doit être suffisamment flexible pour que le contact entre les deux protéines puisse ensuite s'établir [Berg and von Hippel, 1988].

Cette relative non-conservation est caractéristique des séquences impliquées dans la promotion de la transcription de gènes pour lesquels la quantité de produit final obtenue (les protéines) doit être finement régulée. Dans le cas des gènes de *coli* dont les promoteurs sont CAP-dépendants, c'est tout le site de fixation de la CRP, plus les sites situés à -10 et -35 qui sont relativement peu conservés. Si cela n'était pas le cas, c'est-à-dire si les signaux présents dans ces sites étaient très 'forts', le promoteur serait trop efficace et donc difficile à réguler [Botsford and Harman, 1992] [Combrugghe *et al.*, 1984]. Cette situation ne facilite bien sûr pas l'identification de ces sites par une analyse comparative des séquences, elle rend également délicate l'interprétation des résultats obtenus ainsi que nous le verrons plus loin. À cela s'ajoute parfois la difficulté de savoir sur quel brin de l'ADN est localisé le site. Le motif le plus fort n'est pas toujours celui qui se trouve le plus éloigné de la première base transcrite et ne peut servir d'indicateur du sens de lecture. Par ailleurs, dans certains cas, on ignore quel gène est effectivement régulé par la CRP [Combrugghe *et al.*, 1984]. Ces deux aspects, faible conservation qui n'est pas liée à la présence d'erreurs mais est biologiquement nécessaire, et ignorance du sens dans lequel les séquences dont on dispose doivent être 'lues', posent ainsi le problème de l'établissement d'un ensemble de données 'propres' sur lequel tester les algorithmes. Ce problème est très important et est souvent négligé. Il n'est cependant pas toujours facile à résoudre, le cas traité ici en est un exemple.

Comme pour les promoteurs de la transcription dans notre illustration précédente, l'exposition des résultats obtenus va suivre le déroulement chronologique. Nous allons ainsi commencer par travailler sur un ensemble de données établi par Stormo [Schneider *et al.*, 1986] et repris par Lawrence [Lawrence and Reilly, 1990] dans le cadre de son algorithme d'optimisation stochastique (sections 3.4.3.3 et 4.2.2.2). Au départ, ces données ne faisaient que constituer un jeu d'essai particulièrement intéressant pour nos algorithmes car les motifs communs qui correspondent aux sites de fixation de la CRP, et que nous devions être capables de repérer, sont, pour les raisons que nous avons mentionnées, très dégénérés. Ils nous permettaient ainsi de tester au mieux l'efficacité et la robustesse de nos méthodes. Ce n'est qu'ensuite que nous avons été amenés, grâce à A. Danchin [Danchin, 1995], à remettre en question ce jeu d'essai,

à en établir un autre (composé en fait des mêmes séquences mais 'lues' parfois dans un autre sens), puis, en fin de compte, à élaborer l'extension ADN-X décrite dans la section 6.2.1.2. Observons que nous ne sommes concernés ici que par l'identification du site de fixation de la CRP, pas par la localisation des séquences à -10 et -35 des promoteurs correspondants.

Le premier jeu d'essai avec lequel nous travaillons, celui de Stormo, est composé de 17 séquences de 110 bases chacunes sélectionnées dans GenBank (correspondant aux gènes *ara*, *bgl*R mut, *col*E1, *cat*, *crp*, *cya*, *deo*P2, *gal*, *ilv*B, *lac*, *mal*E, *mal*K, *mal*T, *omp*A, *p*BR-P4, *tna*A, *uxu*AB — nous avons éliminé une $18^{ième}$ séquence correspondant au gène *tdc* qui ne provenait pas de GenBank). Ces séquences avaient été découpées par Stormo *et al.* de façon à ce que le motif soit situé à des positions quelconques sur chacune des séquences. Le but des auteurs était de montrer que leur méthode est capable de localiser la position de motifs correspondant à des sites même lorsque les séquences ne sont pas alignées.

Le second jeu d'essai est composé des mêmes 17 séquences, certaines cependant 'lues' en ordre inversé complémentaire et correspondant ainsi à des séquences présentes sur l'autre brin de l'ADN. Ce jeu a été sélectionné par Berg [Berg and von Hippel, 1988], d'après un choix effectué antérieurement par Combrugghe [Combrugghe *et al.*, 1984] auquel 4 séquences ont été ajoutées.

Bien que cela ne soit pas clairement indiqué, nous pouvons supposer que Berg a réalisé le choix du brin sur lequel lire une séquence d'après la séquence qui présentait une plus forte conservation du premier motif $TGTGA$ du site de fixation de la CRP. Quant à Stormo, nous pensons que la sélection qu'il a effectuée (celle de GenBank) correspond plutôt à celle pour laquelle le site est correctement orienté par rapport à la direction de transcription du gène qu'il régule lorsque celui-ci est connu (et unique!). Seule la sélection des sites relatifs aux gènes *mal*K et *p*BR-P4 est douteuse ainsi que nous l'avons vérifié sur la base de données Colibri [Médigue *et al.*, 1993].

Dans tous les cas, la position de ces sites sur chacune des séquences a pu par ailleurs être déterminée expérimentalement. Nous comparons ainsi nos résultats à ces données expérimentales et, lorsque les deux diffèrent, aux résultats obtenus par Lawrence avec sa méthode (celle correspondant à l'article de 1990 [Lawrence and Reilly, 1990]).

6.2.3.2 Résultats

Nous commençons par donner une figure qui permet de mieux comprendre en quoi le problème traité ici est difficile. La figure 6.3 montre un alignement des sites que nous devons être capables d'identifier. Nous ne présentons qu'un site par gène lorsque celui-ci en possède deux. Nous avons, pour le propos de cette figure, choisi celui repéré par l'algorithme de Lawrence (site 2), le choix du site 1 ou un choix arbitraire produirait un résultat analogue. En-dessous de l'alignement, nous indiquons la matrice du nombre des bases observées pour chaque colonne de l'alignement. Nous pouvons voir qu'aucune position parmi les 22 que le site comporte n'est strictement conservée, que sur seulement 3 des 22 positions on observe uniquement 2 bases, et qu'à 11 des 22 positions, toutes les bases sont présentes. Par ailleurs, la même base est présente à plus de 50% dans seulement 13 colonnes de la matrice. Enfin, si nous considérons deux de ces motifs et les comparons entre eux, par exemple le onzième motif (associé au gène *mal*K) et le treizième (associé au gène *omp*A), nous pouvons observer qu'il faut 15 substitutions pour passer de l'un à l'autre, 12 si l'on ne considère que les positions centrales (c'est-à-dire, le motif probablement palindromique et les bases intermédiaires). Notons que la matrice donnée ici est approximativement celle qu'obtient Lawrence [Lawrence and Reilly, 1990] à la fin de son algorithme d'optimisation stochastique, mais nous verrons un peu plus loin qu'il a en fait besoin

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	T	*T*	*T*	*T*	*T*	*T*	*G*	*A*	*T*	*C*	*G*	*T*	*T*	*T*	*T*	*C*	*A*	*C*	*A*	*A*	*A*	*A*
	T	*T*	*A*	*T*	*T*	*T*	*G*	*C*	*A*	*C*	*G*	*G*	*C*	*G*	*T*	*C*	*A*	*C*	*A*	*C*	*T*	*T*
	A	*A*	*C*	*T*	*G*	*T*	*G*	*A*	*G*	*C*	*A*	*T*	*G*	*G*	*T*	*C*	*A*	*T*	*A*	*T*	*T*	*T*
	G	*T*	*A*	*T*	*G*	*C*	*A*	*A*	*A*	*G*	*G*	*A*	*C*	*G*	*T*	*C*	*A*	*C*	*A*	*T*	*T*	*A*
	A	*G*	*G*	*T*	*G*	*T*	*T*	*A*	*A*	*A*	*T*	*T*	*G*	*A*	*T*	*C*	*A*	*C*	*G*	*T*	*T*	*T*
	T	*T*	*A*	*T*	*T*	*T*	*G*	*A*	*A*	*C*	*C*	*A*	*G*	*A*	*T*	*C*	*G*	*C*	*A*	*T*	*T*	*A*
	T	*A*	*A*	*T*	*T*	*T*	*A*	*T*	*T*	*C*	*C*	*A*	*T*	*G*	*T*	*C*	*A*	*C*	*A*	*C*	*T*	*T*
	A	*A*	*A*	*C*	*G*	*T*	*G*	*A*	*T*	*C*	*A*	*A*	*C*	*C*	*C*	*C*	*T*	*C*	*A*	*A*	*T*	*T*
	T	*A*	*A*	*T*	*G*	*T*	*G*	*A*	*G*	*T*	*T*	*A*	*G*	*C*	*T*	*C*	*A*	*C*	*T*	*C*	*A*	*T*
	T	*T*	*C*	*T*	*G*	*T*	*A*	*A*	*C*	*A*	*G*	*A*	*G*	*A*	*T*	*C*	*A*	*C*	*A*	*C*	*A*	*A*
	T	*T*	*C*	*T*	*G*	*T*	*G*	*A*	*A*	*C*	*T*	*A*	*A*	*A*	*C*	*C*	*G*	*A*	*G*	*G*	*T*	*C*
	A	*A*	*T*	*T*	*G*	*T*	*G*	*A*	*C*	*A*	*C*	*A*	*G*	*T*	*G*	*C*	*A*	*A*	*A*	*T*	*T*	*C*
	A	*T*	*G*	*C*	*C*	*T*	*G*	*A*	*C*	*G*	*G*	*A*	*G*	*T*	*T*	*C*	*A*	*C*	*A*	*C*	*T*	*T*
	G	*A*	*T*	*T*	*G*	*T*	*G*	*A*	*T*	*T*	*C*	*G*	*A*	*T*	*T*	*C*	*A*	*C*	*A*	*T*	*T*	*T*
	T	*G*	*T*	*T*	*G*	*T*	*G*	*A*	*T*	*G*	*T*	*G*	*G*	*T*	*T*	*A*	*A*	*C*	*C*	*C*	*A*	*A*
	C	*G*	*G*	*T*	*G*	*T*	*G*	*A*	*A*	*A*	*T*	*A*	*C*	*C*	*G*	*C*	*A*	*C*	*A*	*G*	*A*	*T*
	A	*C*	*C*	*T*	*G*	*T*	*G*	*A*	*C*	*G*	*G*	*A*	*A*	*G*	*A*	*T*	*C*	*A*	*C*	*T*	*T*	*C*
A	6	6	6	0	0	0	3	15	6	4	2	11	3	4	1	1	13	3	12	2	5	4
C	1	1	4	2	1	1	0	1	4	7	4	0	4	3	2	15	1	13	2	6	0	3
G	2	3	3	0	12	0	13	0	2	6	6	3	8	5	3	0	2	0	2	2	0	0
T	8	7	4	15	4	18	1	1	5	2	5	3	2	5	12	1	1	1	1	7	11	9

Tableau 6.3: Sites de fixation des CRP (un seul site sélectionné par séquence) et matrice des bases observées (les lignes corespondent aux symboles A, C, G, et T respectivement) : une indication de la difficulté que pose la localisation de ces sites.

de donner un 'coup de pouce' initial à sa méthode afin qu'elle puisse fournir un tel résultat.

Avec la version originale de H/LeMoivre dont nous disposions à l'époque où nous nous sommes intéressés à ce problème, nous n'étions pas capables de le 'résoudre' car le nombre d'erreurs (même s'il s'agissait uniquement de substitutions) que nous pouvions permettre était limité à 2 ou 3. Comme nous l'avons vu, cela n'est pas suffisant pour repérer ce site. Les versions ultérieures de l'algorithme, en particulier avec la première et la seconde des modifications présentées dans la section 5.4.2.1.5, nous ont cependant permis de travailler avec jusqu'à 6 ou 7 substitutions en un temps qui est long mais demeure acceptable (1 journée pour cet exemple de 17 séquences de longueur moyenne 110 sur une Silicon Graphics Indigo WorkStation (R4000)).

Le premier résultat sur ce jeu d'essai a donc été obtenu en cherchant avec HMoivre les plus longs modèles présents dans au moins 95% des séquences et avec 6 substitutions autorisées au plus. Avec ces paramètres, nous sommes capables d'identifier 14 modèles de longueur 16 présents dans 16 des 17 séquences (la $17^{ième}$ séquence où aucun modèle n'est localisé est associée au gène *cat*). Parmi ces modèles, 2 correspondent au site exactement (ce sont les modèles $\underline{TGTGA}ACTATT\underline{TCACA}$ et $\underline{TGAGATCTAGGTCACA}$ — nous soulignons l'emplacement du motif probablement palindromique). Les 12 autres modèles correspondent au site mais les occurrences sont incorrectes sur une ou plusieurs séquences (jusqu'à un maximum de 6 dans un cas). Par 'incorrecte', nous entendons qu'elles ne représentent pas le site expérimental. Il s'agit parfois du site considéré comme étant le second le plus fort par le modèle statistique de Lawrence. Cela est en particulier le cas pour le site de fixation de la CRP du gène *gal*, qui est presque toujours trouvé 18 positions en amont du site expérimental. D'une manière générale, aucun modèle ne rate le site complètement. Le plus intéressant cependant est que 11 des 14 modèles satisfont en tant qu"étiquette' au motif (hypothétiquement) palindromique $\underline{TGTGA}NNNNNN\underline{TCACA}$ et que les 3 autres le satisfont en grande partie. Enfin aucun autre modèle n'est rencontré. La solution apportée par HMoivre paraît donc être suffisamment

sélective malgré la grande souplesse autorisée dans la définition de ressemblance.

Cette 'solution' est cependant, du point de vue biologique, très contestable pour les raisons indiquées dans l'introduction. Le jeu établi par Stormo (celui utilisé par Lawrence) obéit à un critère de sélection des données, en particulier quant au brin sur lequel il faut lire la séquence, qui n'est pas le seul possible. À la suite d'une suggestion de A. Danchin, nous avons établi un second jeu d'essai dont le critère, correspondant à celui de Combrugghe et de Berg, est le suivant : la séquence où se trouve le site doit être 'lue' sur le brin où le premier motif *TGTGA* apparaît le mieux conservé. L'ensemble obtenu est composé des mêmes séquences que celles du jeu de Stormo, simplement dans 6 cas (séquences associées aux gènes *ara*, *crp*, *deo*P, *gal*, *mal*E et *omp*A) la séquence de Combrugghe est le complémentaire inversé de celle du jeu de Stormo.

Par ailleurs, si le premier jeu d'essai que nous avons réalisé nous a fourni une bonne occasion de tester l'algorithme HMoivre modifié, il nous a aussi montré qu'il serait sans doute plus efficace de travailler, non avec HMoivre, mais avec LeCombi, puisque plusieurs des positions à l'intérieur du site paraissent ne pas intervenir dans la reconnaissance du signal par la protéine CRP et sont donc non-spécifiques.

Pour compléter notre analyse préliminaire, nous avons alors :

- testé une version implémentée par Viari [Viari, 1995] de l'algorithme de Lawrence (version de l'article de 1990) sur le second jeu d'essai (celui de Berg et Combrugghe);
- utilisé LeCombi avec la couverture

$$CCPR = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{A, C, G, T\}, 8), \\ (S, 0)\ \textit{pour tous les autres ensembles} \end{array} \right\}$$

  un quorum de 95% et en autorisant 2 erreurs;
- utilisé ADN-LeCombi (lecture dans les deux sens) dans les mêmes conditions que ci-dessus.

Avec l'algorithme de Lawrence nous avons recherché un motif (c'est-à-dire en fait une matrice) de longueur 22 puisque dans ce cas la taille du site doit être connue à l'avance [Lawrence and Reilly, 1990]. Dans nos algorithmes, nous avons demandé à chaque fois les modèles les plus longs.

Commençons par les résultats obtenus avec l'algorithme de Lawrence. Lorsque nous recherchons avec ce dernier les sites de fixation de la CRP, soit sur les séquences de Stormo, soit sur celles de Berg, nous nous rendons compte qu'en fait l'algorithme ne peut pas vraiment partir de positions initiales quelconques afin de construire la matrice spécifique du site, du moins dans sa version de 1990. Nous rappelons (voir la section 4.2.2.2) que la différence fondamentale entre cette version et celle de 1993 (échantillonnage de Gibbs) porte sur la question du choix du site, que nous noterons 'variable', lorsque les positions de ce site sur les autres séquences sont maintenues fixes (fin de l'étape d'échantillonnage). Dans la méthode d'échantillonnage de Gibbs, la position x sur la séquence 'variable' est choisie avec une certaine probabilité $\frac{L_x}{\sum_x L_x}$ qui dépend à la fois de la quantité d'information L_x que le mot à cette position ajoute à la matrice à une étape donnée, et de celle ajoutée (ou retranchée) par les autres mots de la séquence. Dans la méthode EM de 1990, les positions du site ne sont choisies qu'à la fin. La matrice représente alors les fréquences observées à chaque position du site hypothétique, pondérées par la probabilité que ce site démarre en position x pour tout x (voir la section 4.2.2.2). Si la matrice de départ est trop fausse, la méthode peut rapidement mener à des maximums locaux et produire

des résultats complètement faux, comme nous en avons fait l'expérience. Par contre, il suffit souvent de donner un 'coup de pouce' à cette matrice initiale pour que le résultat devienne bien meilleur. Le coup de pouce que nous lui avons donné a consisté tout simplement à forcer la présence d'un T de manière fortement majoritaire (85%) en quatrième position du site, c'est-à-dire, en première position du motif hypothétiquement palindromique. C'est une stratégie un peu *ad hoc*, et les résultats obtenus que nous présentons maintenant sont donc à prendre avec une certaine prudence. Ces résultats sont :

- par rapport au jeu de Stormo, le site est correctement localisé sur 11 des 17 séquences et manqué sur 6 séquences, correspondant aux gènes *bgl*R, *cat*, *gal* (placé 18 positions en amont comme dans HMoivre), *ilv*B, *mal*T et *omp*A;
- par rapport au jeu de Berg, le site est correctement localisé sur 13 des 17 séquences, il est manqué sur *bgl*R, *cat*, *mal*K et *ilv*B.

Le temps d'exécution de l'algorithme est très court : quelques minutes sur une Silicon Graphics Indigo WorkStation (R4000). Il serait intéressant de voir ce que la version de 1993 (Échantillonnage de Gibbs) est capable de repérer, en particulier s'il est alors possible de partir de n'importe quelles positions initiales comme l'affirment les auteurs.

Quand nous utilisons cette fois LeCombi sur le jeu d'essai de Stormo, nous obtenons un seul plus long modèle (couvrant 17 bases) qui repère le site correctement sur 15 des 17 séquences et le place ailleurs sur la séquence associée au gène *gal*. L'occurrence trouvée dans ce cas se situe à nouveau 18 bases en amont du site expérimental. Rappelons que c'est celle la plus couramment observée avec HMoivre et qu'elle correspond au choix initial de Lawrence dans son article [Lawrence and Reilly, 1990]. L'algorithme de Lawrence ne place le site de *gal* au bon endroit que lorsqu'on y introduit des informations supplémentaires, en particulier concernant la nature hypothétiquement symétrique du site [Lawrence and Reilly, 1990]. Le modèle que nous trouvons, $\underline{GTGA}NNNNNN\underline{TCNCA}NT$, ne reflète plus cette symétrie de manière parfaite. Quant au site qui est 'manqué', il correspond une fois de plus à celui de fixation de la CRP pour la régulation du gène *cat* (il y a en fait deux sites relatifs à *cat*, aucun des deux n'est correctement localisé). Si nous faisons l'hypothèse que ce qu'il fallait identifier dans cet exemple était les sites de la CRP indiqués dans [Lawrence and Reilly, 1990], alors le modèle identifié avec LeCombi paraît légèrement moins 'bon' que les deux meilleurs modèles repérés par HMoivre, cependant c'est le seul que nous trouvons et nous pouvons donc argumenter qu'il est plus sélectif. Par ailleurs, ce modèle est plus long d'une unité que les modèles construits par HMoivre. Il est important de noter également que ces résultats ont été produits en 4 fois moins de temps que pour HMoivre (environ 6 heures sur une Silicon Graphics Indigo WorkStation (R4000)).

Avec le jeu de Berg, nous obtenons, avec des temps sensiblement identiques, 7 modèles, également de longueur 17, qui sont tous associés au site : le premier modèle localise correctement le site sur 16 des 17 séquences, les six suivants placent le site ailleurs sur 1 ou 2 séquences. Seul le modèle $AN\underline{TGTGA}NNTNNN\underline{NCNC}$, dont toutes les occurrences correspondent correctement au site, n'identifie pas ce site sur *cat*. La caractéristique la plus frappante des sept modèles trouvés est qu'ils comportent tous le premier motif $TGTGA$, mais jamais le second complètement, et qu'un T est toujours présent en troisième position après le motif $TGTGA$, à un endroit où il ne semble pas y avoir de symbole majoritaire dans le jeu de Stormo et Lawrence (voir la matrice de la figure 6.3).

Enfin, lorsque nous utilisons ADN-LeCombi sur le jeu de Stormo ou sur celui de Berg (puisque nous lisons les séquences dans les deux sens, le résultat obtenu est le même, quel que soit l'ensemble sélectionné), nous obtenons, en à peu près le double du temps, deux modèles de longueur 18, identiques en termes d'ensembles d'occurrences. Ils repèrent le site correctement

sur 15 des 17 séquences, le manquent complètement sur *cat* et le placent ailleurs sur *gal*. Si nous nous intéressons au 'sens' de lecture maintenant, certains sites sont des occurrences dans les deux sens, ce sont les sites associés à *crp*, *col*E, *lac*, *mal*E et *tna*A. Les autres peuvent être groupés en deux ensembles en termes de leur sens de lecture. Le premier ensemble est constitué des occurrences sur les séquences de *bgl*R, *cya*, *mal*T, *pbr*P4 et *uxu*A. Le second est formé des occurrences sur les séquences de *ara*, *deo*P2, *ilv*B, *mal*K et *omp*A. Observons qu'à l'exception de *mal*K et de *ilv*B, les éléments de ce second ensemble correspondent à des sites qui devraient, selon le critère établi par Combrugghe et Berg, être 'lus' sur le brin opposé de l'ADN par rapport au jeu de Stormo. Selon eux, cela s'appliquerait également aux sites de *mal*E et *crp*. Cependant par rapport aux modèles trouvés ici, le sens de lecture de ces sites nous apparaît plutôt indifférent. Il est par ailleurs intéressant de noter qu'aucun des modèles trouvés ne comporte les deux motifs $TGTGA$ et $TCACA$ exactement. En effet, les deux modèles sont $TA\underline{NGAGA}NNNNNN\underline{TCNCA}$ et $\underline{TGNGA}NNNNNN\underline{TCACN}TA$.

6.2.3.3 Discussion

Les résultats obtenus avec DNA-LeCombi semblent donner une indication quant au brin sur lequel se situerait effectivement le site de fixation de la CRP, mais il faut être prudent quant à leur interprétation. Il est en effet délicat de conclure hâtivement en ignorant l'influence du choix des paramètres sur ces résultats. Ainsi, lorsqu'aucune substitution n'est autorisée (mais toujours 8 jokers) et qu'en conséquence, la longueur des modèles trouvés diminue même pour des quorums bas (par exemple pour $q = 70\%$, cette longueur est de 12), l'orientation des sites suit celle du jeu de Berg simplement parce que le premier motif (le seul 'vu' par l'algorithme dans ce cas) y est toujours très fort. Par contre, lorsque plus de substitutions sont autorisées, et que la longueur maximale des modèles repérés s'allonge, l'orientation peut changer.

Il est important d'observer également que la valeur de certains paramètres a dû être fixée de manière empirique, en particulier celle du quorum. Dans le cas précis de cet exemple, il aurait été plus logique d'établir ce dernier à 100%. C'est ce que nous avions fait initialement. Les résultats obtenus alors sont cependant moins bons car les deux sites associés au gène *cat* sont plus dégénérés et donc difficilement identifiables en même temps que les autres. Exiger leur présence en tant qu'occurrences des modèles construits suffit à diminuer sensiblement la qualité du résultat. Celui-ci dépend ainsi beaucoup d'une certaine homogénéité des données, ou exige que le quorum varie suivant les besoins. Le cas du gène *cat* est intéressant pour une autre raison. Il a été suggéré [Berg and von Hippel, 1988] [Schneider *et al.*, 1986] que deux facteurs peuvent contribuer à cette plus grande dégénérescence apparente des sites de *cat*. Le premier est que ceux-ci sont asymétriques (seul le motif $TGTGA$ est conservé). Le second facteur est que l'espacement entre les deux motifs $TGTGA$ et $TCACA$ serait en réalité de 9 bases au lieu de 6 pour le site primaire et de 7 pour le site secondaire [Berg and von Hippel, 1988]. Nous avons donc voulu relancer nos algorithmes à la recherche de ces sites en permettant cette fois un certain nombre de substitutions et de trous. La difficulté est que, plus nous autorisons de trous entre un modèle et ses occurrences, plus ces occurrences sont dégénérées. Nous en avons ainsi obtenu plusieurs, localisées aux mêmes positions sur chaque séquence mais de longueurs différentes, et nous n'avions pas les moyens de déterminer quelle occurrence était la bonne. Ce problème des trous est en fait plus général que ce qui est observé sur ce simple jeu d'essai. L'autorisation de trous dans un alignement ou un bloc est un problème qui n'a pas, à notre sens, été traité de manière vraiment satisfaisante jusqu'à présent, nous reviendrons sur ce point dans le dernier chapitre. Dans cet exemple en tout cas, comme dans celui des promoteurs dans la section précédente, il est clair qu'il serait plus intéressant de travailler avec le concept d'une

'suite de jokers de longueur variable' localisée entre des motifs sans trou.

6.2.4 Les éléments de régulation par le fer

6.2.4.1 Introduction

Une des fonctions exercées par les protéines consiste à transporter des petites molécules ou des ions à l'intérieur de l'organisme, ainsi par exemple les protéines de la famille des globines avec lesquelles nous travaillerons plus loin interviennent dans le transport de l'oxygène. Le fer est un autre élément essentiel à la vie qui, chez les organismes supérieurs, est transporté par la transferrine dans le plasma du sang et doit former un complexe avec la ferritine afin de pouvoir être stocké dans le foie.

Ces deux protéines, transferrine et ferritine, subissent une régulation de leur expression qui dépend de la quantité de fer présent à un moment donné à l'intérieur des cellules. Cette régulation est traductionnelle, elle a donc lieu au moment de la traduction de l'ARN messager en protéines et non au moment de la transcription.

Nous avons fait observer dans le chapitre 2 que tout le matériel qui est transcrit n'est pas ensuite traduit en protéine, la partie non traduite concerne en particulier la région initiale 5' et la région terminale 3', ainsi que les introns (dans le cas des eucaryotes). Les régions 5' et 3' sont celles qui interviennent dans le processus de régulation de la biosynthèse des protéines au niveau de la traduction. Celui de la ferritine et de la transferrine est ainsi controlé par deux éléments situés dans ces régions (5' pour la ferritine et 3' pour la transferrine). Ces éléments ont été appelés des 'éléments de régulation par le fer'. Ils sont également dénotés IRE de l'anglais 'Iron Responsive Elements'. L'IRE responsable de la régulation de la ferritine est constitué d'une séquence comportant environ 28 nucléotides qui se replie en forme d'épingle à cheveux avec une excroissance vers le milieu de la partie amont de la tige (voir la figure 6.12). La boucle de l'épingle est absolument conservée en termes de séquence, de même que le nucléotide, une Cytosine, formant l'excroissance et qui se situe toujours 5 positions avant le premier nucléotide de la boucle. Le reste de la séquence n'est réellement conservé qu'en tant que structure, du moins entre familles différentes de gènes. La régulation de la transferrine est, quant à elle, réalisée par 5 séquences qui ressemblent à l'IRE de la ferritine.

Ces éléments de régulation sont donc en général recherchés par des programmes permettant de décrire des structures secondaires d'ARN à travers une grammaire qui inclut à la fois des motifs structuraux (pour la tige) et lexicaux (pour la boucle et l'excroissance) [Billoud *et al.*, 1996] [Dandekar *et al.*, 1991] [Dandekar and Hentze, 1995] [Gautheret *et al.*, 1990]. Un exemple d'une telle description dans le cas des IRE est donnée dans la figure 6.12 [Dandekar *et al.*, 1991]. C'est celle employée par le programme de Billoud appelé Palingol.

Notre but avec cet exemple a été au départ de voir dans quelle mesure il n'y avait pas, y compris dans des gènes de familles différentes, une conservation en séquence au niveau de la tige de l'IRE ainsi qu'au niveau des régions situées en amont ou en aval de l'IRE.

Nous avons pour cela travaillé avec deux jeux d'essais parmi les 4 utilisés par Billoud [Billoud *et al.*, 1996]. Les séquences composant ces deux jeux ont été extraites de la banque EMBL version 43 par la base de donnée HOVERGEN [Duret *et al.*, 1994]. Ces deux jeux sont composés de :

- 48 séquences contenant le mot 'ferritin' parmi leurs mots-clés associés à la séquence dans la banque, dont 22 seulement contiennent un vrai IRE (lot A);
- 33 séquences correspondant à des régions 5' et qui comprennent toutes des IRE (certains associés à des ferritines) (lot B).

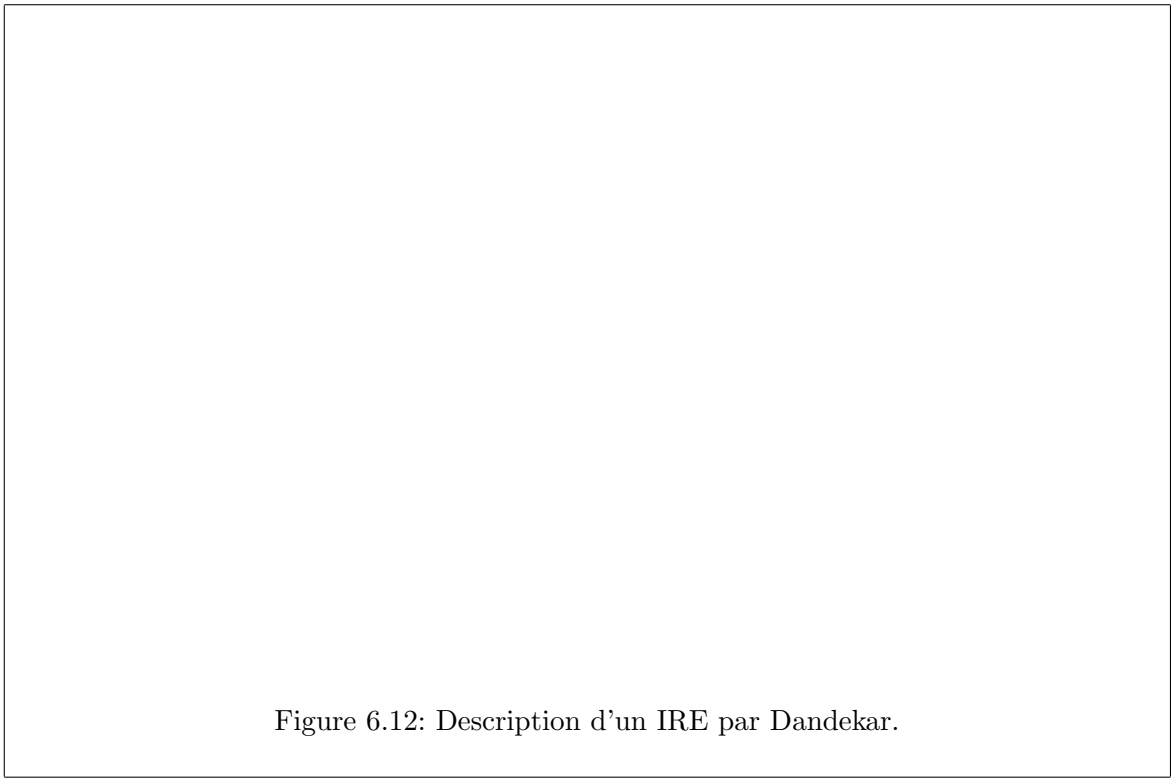


Figure 6.12: Description d'un IRE par Dandekar.

Observons que le programme de Billoud [Billoud *et al.*, 1996] réussit à identifier 19 des 22 séquences du premier jeu qui contiennent des vrais IRE et localise les 33 séquences du second jeu parmi un ensemble plus grand de régions 5' comprennant 17642 séquences en tout. Les trois 'bonnes' séquences du premier jeu qui sont manquées le sont, soit parce qu'elles contiennent une excroissance ou une substitution dans le pied de la tige qui ne sont pas permises dans la description utilisée par Palingol, soit parce que cette partie inférieure de la tige n'est pas suffisamment longue (séquences tronquées).

6.2.4.2 Résultats

Nous avons recherché les modèles les plus longs présents dans les séquences de chacun des deux lots avec HMoivre, Combi (dans sa version originale et la variante qui travaille avec une pondération par cardinalité d'ensemble - section 5.4.2.2.6.2) et LeCombi. Nous avons également fait varier le quorum entre 20% et 70% avec un pas de 10%. Les résultats les plus intéressants ont été obtenus avec Combi (ils sont sensiblement les mêmes avec l'algorithme original ou la variante) et ce sont ceux que nous présentons ici (19 secondes sur une Sparc Station 20).

Nous avons utilisé plusieurs couvertures; une de celles qui se sont montrées les plus 'informatives' sur cet exemple est la suivante :

$$CCP = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{A, C, G, T\}, 0), \\ (S, 1)\ \textit{pour tous les}\ (10)\ \textit{autres ensembles} \end{array} \right\}.$$

De plus, nous n'avons permis que 4 jokers dans la première étape de l'algorithme (variante de la section 5.4.2.2.6.1), ce qui signifie que le nombre total d'ensembles 'spéciaux' (les non

singletons) autorisés dans les modèles était également limité à 4.

Avec un quorum de 40%, les 21 occurrences distinctes des sept modèles de longueur k_{max} égale à 27 identifiés sur le lot A correspondent toutes à l'IRE. Certains de ces modèles peuvent en fait être regroupés en un seul car ils ont exactement les mêmes occurrences et diffèrent uniquement par leurs noms. Les trois modèles obtenus après cette opération sont alors les suivants :

Modèle A1 $C[CT]TG\underline{C}TTCAA\underline{CAGTG}[CT]TTG[AG]ACGGAA[CG]$
(observons que le second ensemble $[CT]$ apparaît comme $[CGT]$ ou $[ACT]$ dans le modèle puisque l'ensemble $[CT]$ n'y est autorisé qu'une fois);

Modèle A2 $C[CT]TG\underline{C}TTCA[AG]\underline{CAGTG}[CT]TTG[AG]ACGGAAC$
(à nouveau ici le second ensemble $[CT]$ apparaît comme $[CGT]$ ou $[ACT]$ dans le modèle et le second ensemble $[AG]$ comme $[AGT]$ ou $[ACG]$);

Modèle A3 $C[CT]TG\underline{C}[GT]TCAA\underline{CAGTG}[CT]TTG[AG]ACGGAAC$
(le second $[CT]$ apparaissant comme $[CGT]$ ou $[ACT]$).

Nous avons souligné les 5 nucléotides de la boucle et le C de l'excroissance qui sont absolument conservés dans tous les IRE.

Ces modèles sont très similaires et peuvent être résumés dans le modèle suivant :

$$C[CT]TG\underline{C}[GT]TCA[AG]\underline{CAGTG}[CT]TTG[AG]ACGGAA[CG].$$

Parmi les ensembles d'occurrences de ces trois modèles (chaque ensemble en comporte 19), 18 occurrences sont communes et représentent 16 IRE identifiés par Palingol ('noms' mnémoniques dans GenBank et positions IRE : HSFERG1 (207-239), HSFERP1 (118-150), HSFERP2 (118-150), HSFERRITH (29-61), HSFHC122 (30-62), MMFERHC (229-261), MMFERHG (967-999), MMFERRH (967-999), OCFERL5 (31-61), RCFERH (24-54), RNFERA1 (205-237), RNFERL1 (400-430), RNFERUTR (20-50), SSFE (12-44), XLFERHSU (7-37) et XLFERRIT (166-198)) ainsi que 2 des 3 IRE ratés par Palingol ('noms' mnémoniques et positions IRE : MMFERLSUB (1198-1213) et MMFERLA (6-21)). Chaque modèle identifie un IRE de plus. Le modèle 1 identifie l'IRE HSAFLP1 (2-28), le modèle 2 l'IRE HSFERHX (949-981) et enfin le modèle 3 l'IRE GGFERH (1306-1336).

Sur le lot B et avec un quorum également de 40%, Combi avec la même couverture CCP que pour le lot A repère initialement 6 modèles de longueur 28 qui représentent deux ensembles d'occurrences distincts et identifient 15 des 33 IRE présents dans le jeu. Ces modèles sont les suivants (après regroupement) (7 secondes sur une Sparc Station 20) :

Modèle B1 $TC[CT]TG\underline{C}TTCA[AG]\underline{CAGTG}[CT]TTG[AG]ACGGAAC$;

Modèle B2 $TC[CT]TG\underline{C}[GT]TCAA\underline{CAGTG}[CT]TTG[AG]ACGGAAC$.

Ils correspondent donc aux modèles 2 et 3 trouvés avec le lot A auxquels un singleton de plus à été ajouté en première position. Sauf pour une des séquences d'origine inconnue (de 'nom' S64727 (482-552)), les IRE correspondants sont tous associés à des ferritines.

La stratégie que nous avons adoptée à partir de ce point sur ce même lot B est, une fois de plus, empirique, mais elle nous a permis d'obtenir des résultats supplémentaires qui peuvent être intéressants. Elle a consisté à retirer du lot B les 15 séquences où le modèle B1 ou B2 est présent, et à relancer Combi à la recherche du plus long modèle présent dans au moins 40% des 18 séquences restantes. Cette seconde recherche a localisé deux modèles de longueur 37 représentant un seul ensemble de 9 occurrences (351 secondes sur une Sparc Station 20). Le modèle après regroupement s'écrit :

Groupe	Modèle	Prés. seq.
1	$TC[CT]TG\underline{C}TTCA[AG]\underline{CAGTG}[CT]TTG[AG]ACGGAAC$	14
	$TC[CT]TG\underline{C}[GT]TCAA\underline{CAGTG}[CT]TTG[AG]ACGGAAC$	14
2	$GAAGGCGTGGCTCCCTC\underline{C}CGGGC\underline{CAGTG}AGCC[CT][GT]G[CG][CG]$	9
3	$C\underline{C}CAGG\underline{CAGTG}CC[CT]TG[GT]G[CT][AG]A$	3
4	$GT\underline{C}[GT][CG]TG[CT]\underline{CAGTG}TGTG[GT]$	2
5	$TCC[AT]\underline{CAGTG}C[AT]GGG[AC][AT]$	2

Tableau 6.4: Modèles trouvés avec Combi sur le lot B de séquences (voir texte pour la couverture CCP et la stratégie suivie — seuls les groupes de modèles ayant au moins deux occurrences sont indiqués).

$$GAAGGCGTGGCTCCCTC\underline{C}CGGGC\underline{CAGTG}AGCC[CT][GT]G[CG][CG]$$

et couvre une partie de l'IRE (environ 3 à 4 bases du côté 3' sont absentes) plus 9 à 12 bases en amont de l'IRE. Il serait intéressant de savoir si cette conservation apparente des séquences avant l'IRE a un lien avec l'activité exercée par ce dernier. Observons aussi que ce modèle est très différent des modèles B1 et B2 caractérisant les ferritines. Les 'noms' mnémoniques des séquences concernées et leurs positions dans le contig d'où elles ont été extraites sont les suivants : HSEP3A1 (1-62), HSEP3A2 (1-62), HSEP3B (1-62), HSEP3C (1-62), HSEP3D (1-62), HSEP3E (1-62), HSEP3F (1-62), HUMPEIR (1-68) et S69200 (1-68). Il s'agit ici essentiellement de gènes codant pour des récepteurs de prostaglandines. Nous avons ensuite renouvelé l'opération, c'est-à-dire que nous avons retiré ces 9 séquences du lot B et relancé Combi dans les mêmes conditions qu'auparavant sur les 9 séquences restantes. Cependant, comme nous ne pouvons plus construire aucun modèle de longueur supérieure à 12, nous avons décidé de baisser le quorum à 30%. Dans ce cas, deux plus longs modèles sont trouvés (en 12 secondes sur une Sparc Station 20), représentant un seul ensemble de 3 occurrences longues de 21 bases et couvrant une partie de l'IRE. Le modèle regroupé s'écrit $C\underline{C}CAGG\underline{CAGTG}CC[CT]TG[GT]G[CT][AG]A$ et est présent dans les séquences HSNMYC (782-853), MMNMYC (1-68) et RATAE3A (3380-3453). Les occurrences du modèle sont identiques sur les deux premières séquences, la troisième sur RATAE3A diffère dans les positions où un ensemble de taille 2 apparaît dans le modèle. Encore une fois, ce modèle ne ressemble à aucun des 2 précédents et il serait intéressant de savoir si le regroupement qu'il opère est fortuit ou non. Parmi les 6 séquences restantes enfin, les IRE ne peuvent plus être regroupés que par des modèles de longueur 18 (2 IRE — modèle $GT\underline{C}[GT][CG]TG[CT]\underline{CAGTG}TGTG[GT]$) et 16 (2 IRE — modèle $TCC[AT]\underline{CAGTG}C[AT]GGG[AC][AT]$). Les IRE des deux dernières séquences sont complètement différents de tous les autres. Tous les modèles trouvés sur le lot B en suivant la stratégie décrite ici se trouvent résumés dans le tableau 6.4.

6.2.4.3 Discussion

Nous ignorons si les résultats indiqués dans la section précédente et la 'classification' primaire qu'ils fournissent des IRE en région 5', ont une valeur biologique quelconque. De notre point de vue cependant, cet exemple est important car seule l'utilisation d'une couverture combinatoire pondérée permet de repérer des caractéristiques susceptibles d'être intéressantes, ce qui n'est pas

le cas des autres techniques dont nous disposons (distance de Hamming entre modèles et mots, couverture pondérée restreinte avec uniquement le joker en plus de l'alphabet des nucléotides). Cependant, nous pouvons également voir à quel point l'information contenue dans les modèles est étroitement liée à la définition de ressemblance adoptée, et au choix des valeurs à attribuer aux divers paramètres. Considérons en effet les trois plus longs modèles identifiés sur le lot A. Ils se ressemblent beaucoup et, ainsi que nous l'avons vu, peuvent être résumés en un seul :

$$C[CT]TG\underline{C}[GT]TCA[AG]\underline{CAGTG}[CT]TTG[AG]ACGGAA[CG].$$

En faisant cela cependant, nous gommons le fait que, par exemple, aucun IRE correspondant à $CCTGCTTCA\underline{G}CAGTGCTTGAACGGAA\underline{G}$ n'est observé parce que ce mot réalise un mélange des modèles A1 et A2 (le premier G souligné provenant du modèle A2, le second du modèle A1). Mais le modèle A1 lui-même a perdu l'information qu'un seul IRE comporte un G en $12^{ième}$ position après le G de la boucle $CAGTG$, et que cet IRE avait une Thymine aux deux endroits où le modèle est composé de l'ensemble $[CT]$ et une Guanine là où il présente un $[AG]$. Est-ce que cette information devait être préservée? La question de la pertinence de cette information ne se rapporte pas seulement aux objets qui sont localisés mais également à leur caractérisation. Cette dernière est essentielle, en particulier si nous voulons pouvoir nous en servir ultérieurement pour identifier d'autres objets du même type. Enfin, une question plus générale concerne les relations de dépendance entre les positions d'une séquence. Pour l'instant nous n'identifions ces relations que par hasard avec nos modèles. Y aurait-il un moyen de formaliser ces relations directement dans les définitions de ressemblance et dans l'étape de recherche des modèles?

Une autre observation à propos de la caractérisation des objets que nous recherchons, dans le cas précis de cet exemple, porte sur une comparaison de celle utilisée par des programmes comme Palingol avec celle que nous obtenons (la liste des modèles trouvés). Rappelons que Palingol n'est pas un programme permettant de déduire une caractérisation à partir d'un ensemble d'exemples (ce que nos algorithmes font d'une certaine façon), mais est un programme qui part d'une description d'un objet et qui recherche les instances vérifiant cette description dans un ensemble de vrais et faux exemples. Il existe cependant une autre différence importante entre notre approche et celle de Palingol qui réside dans le fait que la description que ce dernier utilise est plus riche. Elle fait ainsi intervenir des concepts plus complexes que la simple succession des symboles et peut y compris décrire des objets non encore observés. Ces concepts peuvent également avoir été préalablement extraits d'exemples comme le sont nos modèles. Même dans ce second cas cependant, l'extraction de la description utilisée par Palingol a porté sur des critères purement structuraux et sont différents de ceux qui nous permettent de construire les modèles. Pourrait-on incorporer de tels critères dans nos définitions de ressemblance? Par ailleurs, est-ce que les séquences en tant que telles, et la technique utilisée dans cette illustration en particulier, peuvent apporter quelque chose d'autre?

Enfin, nous nous sommes limités ici à rechercher les modèles les plus longs qui se trouvent présents dans au moins un certain pourcentage des séquences de nos jeux d'essai. Afin de déterminer s'il existe d'autres régions conservées, en particulier en aval de l'IRE, semblables à celle identifiée en amont de quelques-unes des séquences du lot B, il faudrait examiner les modèles ayant une longueur inférieure à la longueur maximale. Une autre façon de procéder serait d'appliquer récursivement l'algorithme non sur des ensembles de plus en plus réduits de séquences comme nous l'avons fait pour ce même lot B, mais sur le lot de départ après avoir masqué les occurrences des modèles trouvés à l'étape antérieure. Cette stratégie générale est celle utilisée par plusieurs heuristiques qui construisent un alignement multiple à partir d'une recherche initiale de blocs de similarité, en particulier [Landraud *et al.*, 1989] [Viari, 1995].

Elle est importante mais présente plusieurs difficultés. La plus importante est sans doute de déterminer sur quel critère appuyer le choix des blocs successivement sélectionnés. Est-ce sur leur longueur ou sur une valeur statistique de leur contenu, et dans ce second cas, comment calculer cette valeur? Ce problème devient d'autant plus aigu que la ressemblance entre séquences, ou entre mots dans les séquences, devient faible. C'est ce que nous allons voir dès la section suivante avec les illustrations portant sur des protéines.

6.3 Protéines

6.3.1 Introduction

L'analyse de séquences protéiques ressemble à celle des séquences d'ADN ou d'ARN et est en même temps très différente sous certains aspects. Elle lui ressemble car le problème central reste la recherche des mots communs dans un ensemble de séquences. Ces mots peuvent être associés aux sites actifs de la protéine et donc à son activité, ou correspondre à des éléments structuraux conservés (et qui, en général, seront alors plus dégénérés au niveau de la séquence). Souvent un site actif est préservé dans sa forme mais une forme conservée peut ne pas être caractéristique d'un site. L'ensemble des formes conservées qui possèdent essentiellement les mêmes positions relatives parmi tous les membres alignés d'une famille est appelé le noyau structural de la protéine (en anglais, le 'core') [Gerstein and Altman, 1995]. Ce noyau représente ainsi une espèce d'armature plus ou moins rigide de la macromolécule.

La première différence entre les deux types d'analyse (ADN/ARN et protéines) est que les définitions de ressemblance s'appliquant aux protéines ne sont pas les mêmes que celles utilisées pour comparer des séquences de nucléotides. En outre, il y a généralement, pour un même ensemble de données, plus de mots 'intéressants' à repérer dans le cas des protéines (celles-ci peuvent comporter de nombreux sites ou motifs structuraux) et ces mots sont en général encore moins faciles à localiser et à distinguer du bruit. Il y a plusieurs raisons à cela. La première est que l'alphabet des protéines est plus grand et que les relations de similarité entre les symboles de cet alphabet sont plus complexes et souvent ambiguës. La seconde est qu'un mot 'intéressant' est en général composé de peu de positions réellement importantes pour la forme ou l'activité de la protéine et qu'en revanche, ce mot possède plusieurs positions non-spécifiques ou hautement variables. Plus souvent encore que dans le cas des séquences nucléiques, nous allons donc devoir faire appel à des techniques de filtrage des premiers résultats obtenus par nos algorithmes.

En fait, dans la plupart des cas, nos algorithmes vont maintenant représenter une première étape d'exploration de l'espace des exemples que nous traitons. Elle sera suivie ensuite par une seconde étape d'évaluation des éléments ramassés lors de la première phase. Observons que cette première étape demeure combinatoire. Ainsi que nous l'avons indiqué dans les chapitres 4 et 5, certaines méthodes, notamment MACAW (section 4.2.2.1.2.2) et l'algorithme de Neuwald (section 5.4.2.2.6.4) mélangent les deux phases : évaluation et recherche des mots communs. Nous pensons qu'il est important de maintenir ces deux étapes séparées. En particulier, la recherche initiale des modèles doit être aussi large et complète que possible, et les propriétés que ces modèles vérifient doivent demeurer claires et précises. Le plus souvent, cela signifie que nous allons conserver plus d'information que nécessaire. La recherche initiale n'est donc pas toujours très sélective, l'essentiel est qu'elle soit sensible, c'est-à-dire, qu'elle 'rate' le moins possible de choses. Dans la seconde phase, il ne nous reste plus alors qu'à déterminer lesquels parmi les modèles obtenus dans la première étape ont une chance d'être réellement pertinents.

Cette seconde étape peut être ajoutée à n'importe lequel de nos algorithmes, à savoir LeCombi, LePoivre ou Klast (dans la suite, nous noterons X cet algorithme). Elle constitue une

extension algorithmique qui demeure à ce jour préliminaire aussi bien au niveau de l'implémentation que des résultats, mais que nous utilisons déjà sous une forme simple dans trois des quatre illustrations présentées dans la suite de ce chapitre. Nous esquissons les idées principales de cette extension dans la section suivante.

6.3.2 Esquisse de la nouvelle extension algorithmique

Cette seconde phase que nous introduisons dans nos algorithmes consiste donc à évaluer la pertinence des modèles trouvés dans la première phase. L'idée sur laquelle nous nous appuyons pour réaliser cela est que, si un modèle localisé par X est effectivement en relation avec un caractère conservé en termes de structure ou de fonction parmi un ensemble de protéines, alors un mot correspondant à ce caractère doit se trouver présent de manière approchée dans toutes ou presque toutes les protéines appartenant à la même famille, ayant la même structure locale, ou exerçant la même activité. Le principe de la méthode se résume donc à repérer avec l'algorithme X les mots communs à un premier ensemble de protéines (i.e. les occurrences d'un même modèle), à associer à chacun de ces ensembles de mots une matrice (qui sera précisée plus loin), puis à calculer le score obtenu par cette matrice lorsqu'elle est successivement alignée avec toutes les positions possibles d'un second ensemble de séquences fonctionnellement proches de celles du premier ensemble. Ce second ensemble peut être réduit à un seul élément. Il peut également être identique au premier. Nous appelons le premier ensemble celui des séquences exemples et le second celui des séquences tests.

Plusieurs méthodes existent pour construire une matrice à partir d'un bloc de mots — on appelle une telle matrice un profil — puis pour rechercher la meilleure occurrence de ce profil sur une ou plusieurs autres séquences. Les profils sont en général établis à partir de mots alignés sans trou, mais leurs occurrences peuvent éventuellement en comporter. La technique de Gribskov [Gribskov *et al.*, 1987] [Gribskov *et al.*, 1988] [Gribskov *et al.*, 1990] autorise des trous dans la phase de recherche (en utilisant la programmation dynamique). Elle peut même partir d'un bloc initial avec trous. Néanmoins, ce qui nous intéresse ici est moins de localiser les occurrences d'un profil que de lui assigner une valeur statistique permettant d'en mesurer la pertinence. Or aucune théorie n'est capable à l'heure actuelle de prendre en compte les trous de manière satisfaisante.

La technique que nous utilisons alors est celle de Waterman, décrite dans [Goldstein and Waterman, 1994] et pour laquelle chaque entrée (i, j) du profil $\mathcal{P}$ obtenu à partir d'un bloc d'occurrences d'un modèle est définie pour tout acide aminé i et position j par :

$$\mathcal{P}(i,j) \quad = \quad \sum_{a=1}^{20} f(a,j) \times \mathcal{M}(i,a) \tag{6.1}$$

avec $f(a, j)$ égal à la fréquence observée de l'acide aminé a à la position j du bloc et $\mathcal{M}$ une matrice de similarité (par exemple, $\mathcal{M}$ peut être la matrice PAM250 de la figure 3.3, ou une des matrices BLOSUM de Henikoff — en général BLOSUM62).

La localisation du meilleur emplacement du profil $\mathcal{P}$ — notons cette position x — sur une séquence s de longueur n, est alors celle ayant le meilleur score S_0 donné par :

$$S_0 = \sum_{j=1}^{k} \mathcal{P}(s_{x+j-1}, j) \tag{6.2}$$

où k désigne la largeur du profil (qui est aussi, dans notre cas, la longueur du bloc et donc celle du modèle). Enfin, la théorie développée par Goldstein *et al.* [Goldstein and Waterman, 1994]

permet d'évaluer la signification statistique de ce score en estimant la probabilité qu'un score supérieur ou égal à S_0 soit observé sur une séquence aléatoire de même longueur et même composition en acides aminés (p-value). La valeur de cette p-value — notée plus simplement p — est égale à :

$$p = 1 - \exp^{-\exp^{-a.(w-c)}} \tag{6.3}$$

avec :

$$a = \sqrt{2.\log(n-k+1)}$$

$$c = a - \frac{\log(\log(n-k+1) + \log(4\pi))}{2a}$$

$$w = \frac{(S_0 - \bar{S})}{\sigma_S}$$

$$\bar{S} = \sum_{j=1}^{k} \sum_{i=1}^{20} f(i) \times \mathcal{P}(i,j)$$

$$\sigma^2_{S_0} = \sum_{j=1}^{k} \sum_{i=1}^{20} f(i) \times \mathcal{P}^2(i,j)$$

et $f(i)$ égal à la fréquence de l'acide aminé i, soit dans la séquence s, soit dans l'ensemble des séquences tests, soit encore dans toute une banque (nous avons opté pour ce dernier choix).

Ceci étant établi, voyons maintenant comment fonctionne notre algorithme en deux étapes. La première, ainsi que nous l'avons dit, consiste à appliquer l'algorithme X (LeCombi ou Klast) au premier ensemble de séquences appelées exemples. Cette application produit un certain nombre de modèles, de longueur fixe ou maximale, présents dans au moins q séquences. Dans la seconde étape, chacun de ces modèles est ensuite considéré à tour de rôle. Pour chacun d'eux, un profil est construit à partir de ses occurrences suivant la formule 6.1 ci-dessus, puis le meilleur emplacement de ce profil est recherché sur chacune des séquences d'un ensemble test. La moyenne des scores obtenus par l'ensemble des meilleurs placements va représenter le score du modèle. Enfin, les modèles sont classés par ordre de p-values croissantes et ceux considérés pertinents sont tous ceux pour lesquels cette p-value est inférieure à une valeur seuil (généralement 10^{-3} ou 10^{-5}).

Lorsqu'un modèle est présent plus d'une fois sur une des séquences, seule est conservée l'occurrence de score maximal. Dans le cas où le quorum de la première étape est strictement inférieur à 100%, un modèle peut ne pas avoir une occurrence sur toutes les séquences. Lors de la seconde étape de l'algorithme, une pseudo-occurrence de ce modèle est alors recherchée sur chacune des séquences s où il n'est pas présent. La position de cette pseudo-occurrence correspond à celle du meilleur emplacement du profil sur s.

Enfin, si un modèle M possède au moins une occurrence qui chevauche l'occurrence d'un autre modèle M', alors seul le modèle ayant la plus petite p-value est conservé.

Nous montrons maintenant l'application de ce nouvel algorithme, que nous avons choisi d'appeler CoSamp-X (de 'Combinatorial Sampling' à partir de l'algorithme X), dans le cadre de trois illustrations représentant des cas considérés difficiles en analyse de séquences : celui des cytochromes P450, celui des globines et enfin celui des protéines contenant un motif structural HTH ('Helix-Turn-Helix'). Dans les trois cas, nous comparons le comportement de CoSamp à celui de CLUSTAL W (section 4.1.3.3.3) et de la méthode d'échantillonnage de Gibbs de Lawrence (section 4.2.2.2) intégrée dans MACAW (section 4.2.2.1.2.2). La première approche, celle de CLUSTAL, est globale et est donc très différente de la notre ou de celle de MACAW mais la comparaison nous a néanmoins semblé intéressante.

6.3.3 Les P450

6.3.3.1 Introduction

Les cytochromes P450 (appelés de manière abrégée les P450) sont des protéines qui catalysent l'oxydation d'une grande variété de substrats hydrophobes et qui sont ainsi impliquées, entre autres, dans le métabolisme des acides gras, des stéroides et des vitamines et dans l'élimination de substances exogènes toxiques. Ces protéines possèdent également une action négative puisqu'elles peuvent exercer un rôle dans l'activation pro-carcinogène [Hasemann *et al.*, 1995] [Stryer, 1981].

Comme pour beaucoup d'autres protéines intervenant dans les processus d'oxydation, les P450 contiennent un groupe hème, qui est un groupement prosthétique composé d'une partie organique et d'un atome de fer qui se lie à l'un des atomes de la chaîne polypeptidique. Dans le cas des P450, contrairement à la plupart des autres protéines qui possèdent un hème, cet atome est le soufre d'une cystéine (C) (les P450 sont ainsi appelés des protéines hème-thiolate). L'acide aminé auquel se lie le groupe hème s'appelle le ligand proximal du groupe.

Les P450 sont regroupés en familles et en sous-familles. Une famille est constituée des P450 présentant une similarité de séquence supérieure à 40%. On définit une sous-famille lorsque cette identité atteint 55% [Nelson *et al.*, 1993]. À l'heure actuelle, on compte 59 familles et 105 sous-familles. Les P450 avec lesquels nous allons travaillé ici sont des éléments isolés n'appartenant à aucune famille précise (ils ont moins de 20% de résidus identiques lorsque alignés optimalement par paires). En termes de séquences, ces protéines se trouvent donc dans ce que Doolittle a désigné la 'twilight zone' [Doolittle, 1981] [Doolittle, 1986], c'est-à-dire qu'elles forment un groupe de protéines probablement homologues et exerçant des activités similaires qui toutefois possèdent moins de 25% de résidus identiques.

Jusque vers la fin 1995, seules trois P450, tous bactériens, avaient pu être cristallisés. Ce sont les $P450_{Cam}$, $P450_{Terp}$ et $P450_{BM_3}$. Ce nombre est actuellement de quatre, le quatrième étant le $P450_{Eryf}$ également bactérien. Des analyses 'manuelle' [Hasemann *et al.*, 1995] et automatique [Jean *et al.*, 1996] des structures des trois premiers P450 cristallisés ont révélé un certain nombre d'éléments structuraux, c'est-à-dire de motifs, relativement bien conservés, alors que les mots lexicaux correspondant à ces motifs sont très dégénérés. Ces éléments structuraux conservés peuvent être groupés de manière parfois différente selon la méthode utilisée pour les repérer. Nous suivons ici la classification donnée dans [Jean *et al.*, 1996] et obtenue avec l'algorithme KMRC/S adapté aux structures (voir section 4.2.3.5) ainsi que celle de [Hasemann *et al.*, 1995] là où il y a divergence. Treize motifs sont ainsi identifiés, dont deux sont en fait composés de deux blocs distincts. La figure 6.13 montre un alignement des trois premiers P450 cristallisés et l'emplacement des blocs structuraux correspondant aux motifs structuraux trouvés sur les trois premiers par KMRC/S. La figure 6.14 montre ces motifs structuraux superposés et la figure 6.15 montre leur arrangement dans l'espace pour le $P450_{Eryf}$. Enfin, le tableau 6.5 fournit la liste des motifs structuraux et les positions qu'ils recouvrent pour le $P450_{Cam}$, le $P450_{Terp}$, le $P450_{BM_3}$ et le $P450_{Eryf}$. Là où il y a désaccord entre P. Jean et Hasemann à propos de ces positions (hélices F et G), nous indiquons les deux. Observons que ces blocs structuraux correspondent souvent à des assemblages de structures secondaires, et l'alignement des séquences montre bien que celles-ci sont très peu conservées entre les $P450_{Cam}$, $P450_{Terp}$ et $P450_{BM_3}$. En effet, seules 26 positions, parmi les 255 que couvre l'ensemble des blocs, présentent le même acide aminé dans les trois P450 (parmi ces 26, 8 se situent dans les blocs 12A et 12B de la région de l'hème, y compris la cystéine où se fixe l'atome de fer). Ces éléments structuraux eux-mêmes ne possèdent pas tous le même degré de conservation. Les hélices G et surtout F sont plus variables. Jean *et al.* ont ainsi décidé de ne pas con-

Bloc	Él^t. de structure	Position sur P450$_{BM3}$	Position sur P450$_{Cam}$	Position sur P450$_{Terp}$	Position sur P450$_{Eryf}$	Long. du bloc
1	β_{1-2} + B	48-62	61-75	50-64	38-52	15
2A	C	94-111	106-123	108-125	96-113	18
2B	D	112-129	124-141	126-143	114-131	18
3	E	144-156	154-166	157-169	144-156	13
4	F	171-185 (J) 174-184 (H)	171-185 (J) 174-184 (H)	174-188 (J) 180-190 (H)	(indéterminé)	15 (J) 11 (H)
5	G	205-225 (J) 201-223 (H)	193-213 (J) 193-215 (H)	212-232 (J) 208-230 (H)	185-205	21 (J) 23 (H)
6	H	228-239	214-225	233-244	206-217	12
7	β_{5-2} + I + J	248-292	232-276	251-295	225-269	45
8	K + β_{6-1}	312-328	279-295	298-314	272-288	17
9	β_{1-4} + β_{2_1}	330-343	296-309	316-329	290-303	14
10	β_{2-2} + β_{1-3} + K'	345-367	310-332	330-352	304-326	23
11	Meander	370-377	334-341	354-361	328-335	8
12A	Cys-pocket	390-398	347-355	367-375	341-349	9
12B	L	399-416	356-373	376-393	350-366	18
13	β_{4-2} + β_{3-2}	441-449	399-407	418-426	395-403	9

Tableau 6.5: Blocs structuraux trouvés par P. Jean [Jean *et al.*, 1996] sur les P450. Là où il y a désaccord avec Hasemann, les deux positions sont indiquées (notées J et H respectivement). Une lettre majuscule pour les éléments structuraux dénote une hélice, β un feuillet.

sidérer la région correspondant à l'hélice F dans leur travail de modélisation de la structure du P450$_{Eryf}$ (non cristallisé à l'époque) à partir de l'alignement multiple structural et en séquence des P450$_{Cam}$, P450$_{Terp}$ et P450$_{BM_3}$ [Jean *et al.*, 1996].

Certains de ces motifs structuraux sont étroitement associés à l'activité des P450, ainsi par exemple, la région de la cystéine (également appelé 'Cys-pocket' parce qu'elle forme une poche ou cavité) dans laquelle se trouve placé l'hème, les hélices F, G et H qui se trouvent proches de la surface de la macromolécule et dont on pense qu'elles contiennent le site de reconnaissance du substrat, la région appelée 'meander' proche de la 'Cys-pocket' qui exerce probablement un rôle dans les réactions d'oxydation et de réduction des électrons venant s'attacher à l'hème (en anglais 'redox-partner protein interaction'), enfin la longue hélice centrale I qui possède sans doute une fonction d'activation de l'oxygène moléculaire pour l'hème.

Notons finalement que tous ces motifs structuraux ont également été retrouvés de manière pratiquement identique (la longueur des blocs est parfois une ou deux unités plus petite) sur le P450$_{Eryf}$, plus récemment cristallisé. L'alignement des quatre P450 cette fois (figure 6.16) montre que la conservation en termes de séquence au niveau de ces blocs est encore plus faible : seules 15 positions présentent le même acide aminé, dont 7 dans les blocs 12A et 12B ('Cys-pocket' et hélice L). Dans ce cas, un bloc structural de plus est trouvé par P. Jean, entre les blocs 1 et 2A précédemment identifiés sur les trois premiers P450 cristallisés.

L'intérêt que présentait pour nous ces protéines était justement de déterminer dans quelle mesure nos algorithmes sont capables de repérer une partie au moins de ces motifs structuraux, en ne considérant que les séquences, c'est-à-dire en ignorant les informations structurales. Cette étude était une bonne occasion également pour examiner et comparer le comportement des deux principales familles d'algorithmes destinés à l'analyse des protéines que

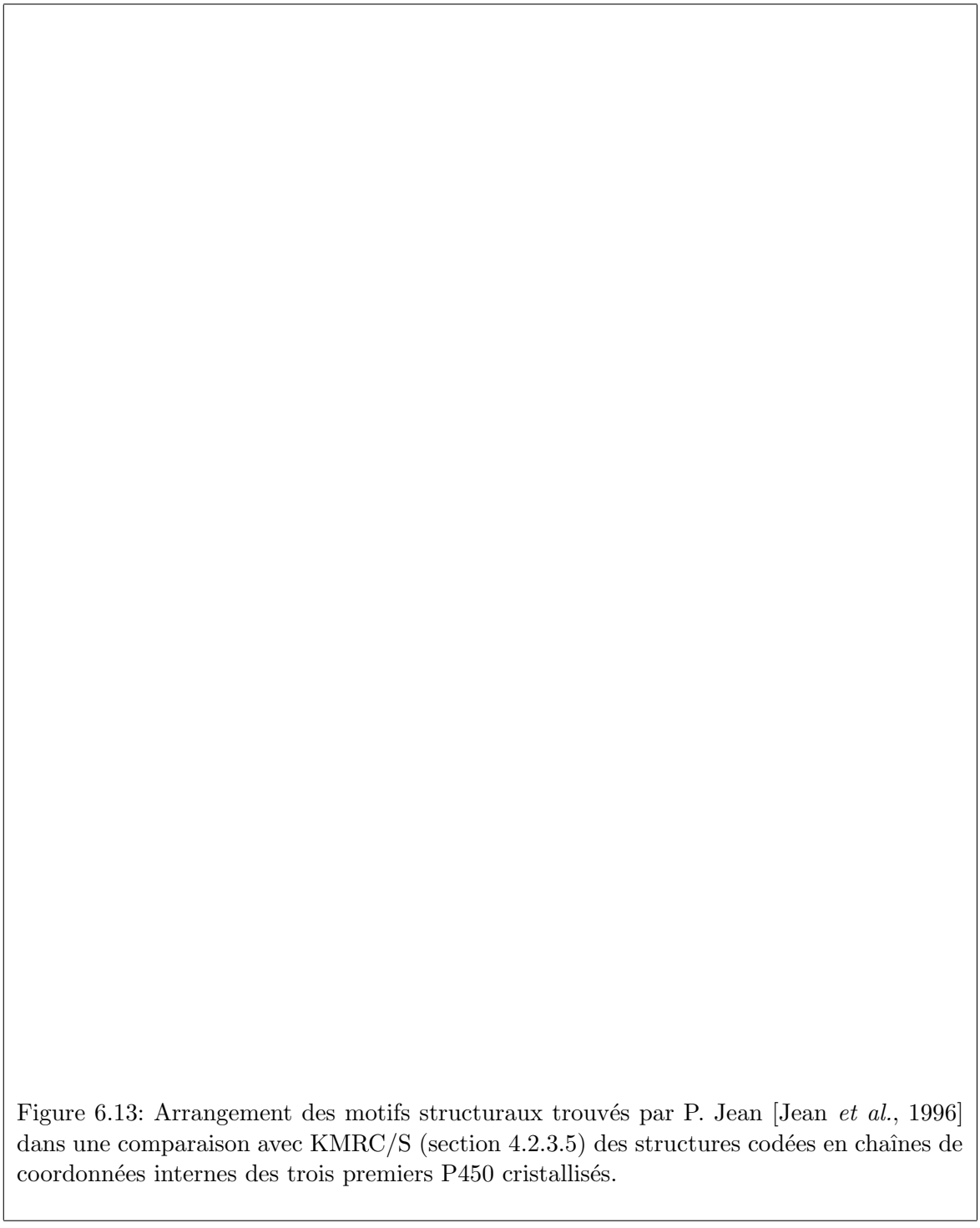


Figure 6.13: Arrangement des motifs structuraux trouvés par P. Jean [Jean *et al.*, 1996] dans une comparaison avec KMRC/S (section 4.2.3.5) des structures codées en chaînes de coordonnées internes des trois premiers P450 cristallisés.

Figure 6.14: Motifs structuraux identifiés sur les $P450_{Cam}$, $P450_{Terp}$ et $P450_{BM_3}$ par KMRC/S [Jean *et al.*, 1996]. Ces motifs sont montrés superposés et correspondent (de la gauche vers la droite et du haut vers le bas) aux blocs 1, 2A, 2B, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12A, 12B, 13.

Figure 6.15: Structure du P450$_{Eryf}$.

Figure 6.16: Arrangement des motifs structuraux trouvés par P. Jean [Jean *et al.*, 1996] dans une comparaison avec KMRC/S (section 4.2.3.5) des structures codées en chaînes de coordonnées internes des quatre premiers P450 cristallisés.

nous avons développés : la famille LePoivre-LeCombi travaillant avec une notion de distance et dont l'unité de comparaison est l'ensemble des symboles de l'alphabet des acides aminés, et Klast travaillant avec une mesure de similarité et dont l'unité de comparaison est l'ensemble des sous-mots de longueur w.

Nous avons pour ce faire utilisé deux jeux d'essais comme ensembles de séquences exemples. Le premier est composé des $P450_{Cam}$, $P450_{Terp}$ et $P450_{BM_3}$ (séquences extraites des entrées PBD [Bernstein *et al.*, 1977] 3cpp, 1ept et 2hpd). Le second est composé de ces trois mêmes séquences plus celle du dernier P450 cristallisé, le $P450_{Eryf}$ (séquence extraite de l'entrée PDB 1oxa). L'ensemble de séquences tests est constitué, soit du $P450_{Eryf}$ tout seul, soit d'un P450 humain de la famille III considérée comme étant la plus proche des P450 bactériens en termes de séquence (de nom CP34_HUMAN dans SWISS PROT [Bairoch and Boeckmann, 1991]), soit d'un jeu composé de tous les 9 P450 humains de la famille III (de noms CP31_RAT, CP32_RAT, CP33_HUMAN, CP34_HUMAN, CP35_HUMAN, CP36_RABIT, CP37_HUMAN, CP38_MACFA, CP3C_CANFA dans SWISS PROT), soit enfin de l'ensemble des quatre P450 cristallisés.

6.3.3.2 Résultats

Nous présentons maintenant quelques-uns des résultats préliminaires que nous avons obtenus sur ces séquences, tout d'abord ceux obtenus avec CoSamp, puis ceux produits par CLUSTAL W et MACAW + Gibbs.

Les résultats peuvent être rangés en trois groupes comportant chacun un ou deux essais. Dans tous les cas, la figure correspondant à chacun des essais montre les modèles trouvés par p-value croissante. Ces figures indiquent :

- dans une première ligne, le score et la p-value associée au modèle (i.e. celle du profil construit à partir de son ensemble d'occurrences) ainsi que l'élément structural concerné (numéro du bloc dans l'article de P. Jean et dans le tableau 6.5 et nom de la structure secondaire);
- le bloc (en séquence) correspondant au modèle avec, dans chaque cas, le nom des P450, les positions et les occurrences elles-mêmes.

Lorsqu'une de ces occurrences est mal placée sur une des séquences, nous signalons à côté de celle-ci le décalage observé par rapport à la bonne position, ou marquons simplement 'faux' si ce décalage est trop grand (plus de 10 positions). La position considérée comme correcte est celle établie par Jean sauf dans deux cas, clairement indiqués, où les positions du bloc lexical que nous trouvons correspondent à celles des blocs structuraux de Hasemann.

Par ailleurs, les algorithmes, paramètres, ensembles d'exemples et tests avec lesquels nous avons travaillé ont été :

Groupe 1

- Séquences exemples : $P450_{BM3}$, $P450_{Cam}$, $P450_{Terp}$
- Séquence test : $P450_{Eryf}$
- Algorithme utilisé en première phase : Klast dans sa version 'seuil sur la moyenne des scores des sous-mots' (section 5.4.3.5.2)
- Paramètres :
  - quorum : 100%

- matrice : PAM250 normalisée (figure 3.7)
- seuil t : 62
- longueur des sous-mots w : 3
- longueur des modèles : 10

- Temps d'exécution de Klast : 379 secondes (sur une Silicon Graphics Indigo Work-Station (R4000))
- Nombre de modèles initiaux : 26668 correspondant à uniquement 196 ensembles distincts d'occurrences

Groupe 2

- Séquences exemples : $P450_{BM3}$, $P450_{Cam}$, $P450_{Terp}$
- Séquences tests :

  essai 1 $P450_{Eryf}$
  essai 2 neuf P450 III humains
- Algorithme utilisé en première phase : LeCombi
- Paramètres :
  - quorum : 100%
  - nombre de jokers : 8
  - nombre de substitutions autorisées : 1
  - longueur des modèles : 15
- Couverture pondérée restreinte :

$$CCPR = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in C, \\ (\Sigma, 8), \\ (S, 0)\ \textit{pour tous les autres ensembles} \end{array} \right\}$$

  avec :

$$C = \left\{ \begin{array}{ll} \{I, L, M, V\} & (\textit{hydrophobe}), \\ \{A, G\} & (\textit{très petit}), \\ \{A, S, T\} & (\textit{petit}), \\ \{C\} & (\textit{cystéine}), \\ \{F, Y, W\} & (\textit{aromatique}), \\ \{K, H, R\} & (\textit{basique}), \\ \{E, D\} & (\textit{acide}), \\ \{N, Q\} & (\textit{glutamine et glutamate}), \\ \{G, P\} & (\textit{apparaît dans les structures contraintes}) \end{array} \right\}$$

- Temps d'exécution de LeCombi : 3843 secondes (sur une Sparc Station 20)
- Nombre de modèles initiaux : 188906 correspondant en fait à 15452 ensembles distincts d'occurrences

Groupe 3

- Séquences exemples : $P450_{BM3}$, $P450_{Cam}$, $P450_{Terp}$, $P450_{Eryf}$

- Séquences tests :

  essai 1 les mêmes séquences exemples

  essai 2 le P450 humain CP34_HUMAN
- Algorithme utilisé en première phase : LeCombi
- Paramètres :
  - quorum : 75%
  - nombre de jokers : 6
  - nombre de substitutions autorisées : 1
  - longueur des modèles : 14
- Couverture pondérée restreinte : la même que pour le Groupe 2
- Temps d'exécution de LeCombi : 3275 secondes (sur une Sparc Station 20)
- Nombre de modèles initiaux : 128622 modèles correspondant à 4281 ensembles d'occurrences distincts

Notons que les p-values de la figure 6.20 sont très basses relativement à celles observées avec les autres essais. Nous suggérons comme explication à ce comportement le fait que les séquences tests sont ici les mêmes que les séquences exemples. Cela pose d'ailleurs la question de savoir comment traiter un tel choix dans la seconde étape, de telle sorte que ce phénomène ne se produise pas. Observons que tous les modèles repérés avec cet essai sont présentés, sauf trois ayant des p-values légèrement plus élevées et qui sont complètement faux. Aucun autre modèle n'est identifié.

Les résultats obtenus avec CLUSTAL W et MACAW + Gibbs sont quant à eux donnés dans les figures 6.22 et 6.23. Dans les deux cas, le jeu d'essai utilisé est composé des $P450_{Cam}$, $P450_{Terp}$ et $P450_{BM3}$. Comme CLUSTAL est un algorithme d'alignement global et non un programme de recherche de blocs, ceux indiqués sur la figure (correspondant à des blocs structuraux qu'il aligne correctement) ont été ajoutés par nous *a posteriori*. CLUSTAL W aussi bien que MACAW + Gibbs ont été utilisés avec leurs paramètres par défaut, sauf dans le cas de MACAW + Gibbs où les longueurs minimale et maximale des blocs recherchés doivent être fixées par l'utilisateur. Nous les avons établies à 4 et 18 respectivement.

6.3.3.3 Discussion

Considérons d'abord les résultats fournis par CoSamp.

La première observation que nous pouvons faire est que ces résultats sont d'inégales qualités selon l'algorithme X, les paramètres utilisés en première phase et les ensembles employés en exemple et en test. Le 'meilleur' est celui de la figure 6.18 pour deux raisons. La première est que cet essai localise partiellement 8 des 13 blocs structuraux, y compris la région du 'meander'. La seconde est qu'il ne présente aucun modèle totalement faux parmi ceux dont la p-value est inférieure à 10^{-6}. De plus, pour les trois derniers modèles qui possèdent une p-value inférieure à 10^{-3}, la valeur de celle-ci chute brutalement. Dans les autres essais, cette valeur, soit décroît de manière continue, soit chute bien plus tôt alors que des 'bons' modèles existent encore.

Il nous est difficile pour l'instant d'expliquer pourquoi le comportement de CoSamp varie ainsi avec les données d'entrée et avec la méthode employée en première phase. L'algorithme lui-même, et surtout sa seconde partie, a certainement besoin d'être plus profondément réfléchi. Une première amélioration qui pourrait lui être apportée concerne les modèles éliminés parce qu'une de leurs occurrences chevauche celle d'un autre modèle ayant une plus petite p-value. Cette élimination ne devrait pas avoir lieu lorsque l'ensemble des occurrences d'un des modèles

Figure 6.17: Résultat première série P450 (Klast, trois séquences, quorum 100%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

Figure 6.18: Résultat seconde série — premier essai P450 (LeCombi, trois séquences, quorum de 100%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

Figure 6.19: Résultat seconde série — deuxième essai P450 (LeCombi, trois séquences, quorum de 100%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

Figure 6.20: Résultat troisième série — premier essai P450 (LeCombi, quatre séquences, quorum de 75%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

Figure 6.21: Résultat troisième série — second essai P450 (LeCombi, quatre séquences, quorum de 75%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

Figure 6.22: Alignement multiple global obtenu avec CLUSTAL W sur les trois premiers P450 cristallisés. L'indication des blocs structuraux correctement alignés a été réalisée par nous, *a posteriori*. Une étoile en-dessous d'une colonne indique la présence de résidus identiques à cet endroit de l'alignement, un point la présence de résidus équivalents.

Figure 6.23: Blocs trouvés par la méthode d'échantillonnage de Gibbs sur les trois premiers P450 cristallisés.

représente un simple décalage de l'ensemble des occurrences de l'autre : il est clair qu'alors un des deux modèles est un prolongement de l'autre et tous les deux devraient être conservés dans la liste finale. Ce cas n'est pas rare, et il risque d'être d'autant plus fréquent que certains des blocs structuraux sont longs par rapport à la longueur des modèles recherchés. Cela est le cas en particulier ici où la longueur des motifs structuraux varie entre 8 et 23.

Cette variation dans la longueur des blocs structuraux est sans doute un des facteurs qui font que certains blocs ne sont pas repérés en tant que blocs lexicaux. Considérons le cas des essais des groupes 2 et 3. La longueur des modèles recherchés par LeCombi est de 15 et 14 respectivement. Cela nous donne assez peu de chances de localiser les blocs 1, 2B, 3, 4, 6, 9, 11 et 13 sauf s'ils sont collés à un autre bloc comme cela est le cas de la 'Cys-pocket' (bloc 12A) qui est juxtaposée au bloc 12B ce qui lui permet, outre sa plus forte conservation, d'être aisément identifiée. Ces blocs courts correspondent effectivement à ceux qui ne sont jamais, ou presque jamais, repérés bien que dans certains cas (e.g. blocs 1, 2B, 4), cela puisse être plutôt dû au fait qu'ils sont très variables en séquence (ils le sont également en structure).

L'argument que nous venons d'avancer expliquant pourquoi certains blocs ne sont pas trouvés paraît moins valable lorsque l'on considère l'essai 1 réalisé avec Klast pour lequel la longueur des modèles recherchés est égale à 10. En effet, malgré cette plus petite longueur, Klast ne parvient pas à localiser certains blocs structuraux, comme les blocs 1, 3, 6 et 9 qui sont maintenant 'à sa portée'. Cet 'échec' est cependant lié à une faiblesse qui est intrinsèque à Klast et n'invalide pas le raisonnement sur la longueur des modèles. Le résultat obtenu avec Klast est en effet moins bon dans la mesure où des modèles totalement faux se trouvent mélangés, parfois avec une p-value petite, aux modèles corrects ou partiellement corrects. Ce phénomène est également observé pour l'essai de la figure 6.21, mais la raison, dans ce cas, réside plutôt dans le choix de l'ensemble test.

Ce choix semble en effet être important pour la qualité des résultats. Ce n'est pas le seul, et la difficulté principale dans l'utilisation de CoSamp (que ce soit avec LeCombi ou Klast) demeure dans la sélection de la méthode à employer en première phase, le choix des paramètres à fixer pour cette méthode (en particulier longueur des modèles et quorum) et de l'ensemble des séquences exemples et tests. Notons par ailleurs que l'évaluation statistique des modèles peut être plus difficile à établir lorsque ceux-ci sont plus courts. Il est ainsi sans doute un peu prématuré pour l'instant de conclure que le modèle de ressemblance de LeCombi est plus approprié que celui de Klast au problème que nous voulions résoudre ici.

Nous voulons également attirer l'attention sur un phénomène observé deux fois dans cet exemple des P450 (et qui le sera encore plus dans le suivant) et qui concerne la question des hélices. À deux reprises en effet, une occurrence se trouve décalée de +3 par rapport au bloc structural que le modèle identifie. Un tel décalage de 3 ou 4 positions correspondant à un tour d'hélice par rapport à sa position correcte n'est pas observé uniquement lorsque nous essayons de trouver des blocs dans les séquences. Les algorithmes qui recherchent des motifs 3D répétés de manière automatique produisent souvent de tels décalages. Ceux-ci doivent alors être 'corrigés' *a posteriori* par une inspection visuelle des structures superposées.

Voyons maintenant les résultats obtenus par les deux autres méthodes. Celle de l'échantillonnage de Gibbs intégré dans MACAW n'est capable de repérer correctement que le bloc 12A correspondant à la 'Cys-pocket' ainsi que le début de l'hélice L. Tous les autres blocs construits sont faux, et d'ailleurs donnés comme étant statistiquement non 'significatifs'. Un de ces blocs est montré dans la figure 6.23. Il est intéressant de noter que, lorsque nous recherchons avec Klast ou LeCombi seulement (c'est-à-dire sans la seconde étape de CoSamp) les plus longs modèles présents à 100% dans l'un ou l'autre des jeux d'essais que nous avons utilisés ici, ce sont justement ces blocs que nous retrouvons toujours et exclusivement.

Le résultat produit par CLUSTAL W est quant à lui un peu surprenant. Comme nous l'avons mentionné, CLUSTAL ne cherche pas des blocs, mais réalise un alignement global des séquences. Cet alignement établit toutefois une mise en correspondance correcte des blocs 5, 6, 7, 8, 9, 10, 12A, 12B et 13 des $P450_{Cam}$, $P450_{Terp}$ et $P450_{BM3}$ ainsi que nous avons pu le vérifier manuellement. L'alignement obtenu avant le bloc 5 est, en termes de structure, incorrect. Cette partie N terminale de la protéine correspond en effet à celle de moindre conservation en séquence aussi bien qu'en structure. Les blocs lexicaux que nous y repérons nous-mêmes sont toujours faux sur une des séquences (en général sur le $P450_{BM3}$) ou, dans un cas, décalé d'un tour d'hélice (bloc 5, hélice 6 sur la figure 6.17). La question que nous pouvons nous poser par rapport à ce résultat de CLUSTAL W concerne une observation que nous avions faite dans le chapitre 4 à propos des méthodes globales par rapport aux locales. Nous avions indiqué alors (section 4.2.1) que certains auteurs (notamment Posfai et Barton) considèrent important de comparer les séquences globalement même lorsque ce sont les similarités locales qui sont recherchées. Cette expérience semble être un argument en faveur de cette opinion. L'illustration de la section suivante le sera également, mais nous verrons avec les motifs 'Helix-Turn-Helix' que l'approche défendue par Posfai et Barton peut ne pas être appropriée dans tous les cas de figure.

6.3.4 Les globines

6.3.4.1 Introduction

Les hémoglobines et les myoglobines (abrégées en globines) sont les principaux transporteurs d'oxygène à l'intérieur des cellules des vertébrés, la première protéine circulant dans le sang et la seconde étant localisée dans les muscles. La capacité des deux protéines à fixer l'oxygène dépend de la présence d'un groupe hème dont le ligand proximal est cette fois une histidine (H) et non plus une cystéine comme dans le cas des P450.

Comme pour les P450, une étude des globines qui ont pu être cristallisées a montré que celles-ci sont très conservées en termes de structure alors que les séquences ne présentent qu'une très faible ressemblance entre elles. En effet, deux acides aminés seulement sont connus pour être absolument conservés dans toutes les séquences (dont l'histidine liée à l'hème), et le pourcentage de résidus identiques entre paires de séquences alignées n'excède en moyenne pas 16% [Bashford *et al.*, 1987] [Lesk and Chothia, 1980]. En 1987, Bashford *et al.* ont réalisé une étude de la variation des acides aminés situés aux positions correspondant aux éléments de structure secondaire conservés dans un jeu de 7 globines (celles qui avaient pu être cristallisées à l'époque). Ces éléments correspondent à 8 hélices notées de A à H. À partir de cette étude, ils ont pu construire deux 'super-motifs', composés chacun d'une suite de six mots lexicaux. Chacun de ces mots est constitué soit des acides aminés réellement observés à ces positions, soit des acides aminés qui, bien que non observés, auraient pu se placer à une de ces positions sans que la structure sans trouve modifiée parce qu'ils sont compatibles en termes de taille (volume), caractère hydrophobe et polarité. Ils ont ensuite criblé la banque de séquences disponibles (composée alors de 3512 séquences dont 226 globines) en recherchant chacun de ces deux 'super-motifs' avec une méthode similaire à celle des profils de Gribskov (sans trou) [Gribskov *et al.*, 1988] [Gribskov *et al.*, 1990] [Gribskov *et al.*, 1987] ou de Waterman [Goldstein and Waterman, 1994] que nous utilisons dans CoSamp-X. Dans leur cas cependant, le profil est constitué d'une suite de profils. Ils ont pu montrer ainsi que les 'super-motifs' permettaient de repérer les 226 globines connues avec un très faible nombre de faux-positifs. Ces faux-positifs sont pour la plupart des polyprotéines qui, à cause de leur longueur et de leur nature, finissent par ressembler à presque tout. La construction de ces 'super-motifs' dépendait cependant de la réalisation d'un alignement préalable des séquences, qui lui-même

était obtenu à partir d'un alignement manuel des structures. Ultérieurement, de tels alignements ont pu être réalisés de manière quasi-automatique [Gerstein and Altman, 1995] ou automatique [Escalier *et al.*, 1996], soit sur les même sept globines de Bashford [Escalier *et al.*, 1996], soit sur ces globines plus une huitième cristallisée après 1987 [Gerstein and Altman, 1995]. Les deux algorithmes permettent ainsi de localiser les fragments de structure tertiaire des globines les plus conservés et qui forment le noyau structural de ces protéines. Les résultats obtenus par Gerstein et Escalier sont essentiellement les mêmes, sauf en ce qui concerne l'hélice appelée F qui fixe l'hème et que Gerstein estime non conservée. Il considère en effet que les résidus composant cette hélice n'occupent pas toujours strictement les mêmes positions relatives par rapport aux résidus des autres éléments structuraux du noyau lorsque les huit globines de son jeu sont mises en correspondance. Par contre, Gerstein aussi bien qu'Escalier considèrent que l'hélice D ne fait pas partie du noyau.

Encore une fois, notre but avec cet exemple a été de voir si nos algorithmes étaient capables de localiser les mots lexicaux correspondant aux motifs structuraux du noyau à partir des séquences uniquement et sans alignement préalable de celles-ci. La méthode employée a été la même que pour les P450. Cette fois cependant, le jeu d'essai comprennait plus de séquences puisque nous avons choisi comme ensemble d'exemples six des sept globines de Bashford (séquences extraites des entrées PDB [Bernstein *et al.*, 1977] 2lhb, 2hgb et 1ecb (hémoglobines), 1mdb (myoglobine) et 2hhb (hémoglobines, chaînes A et B)) et utilisé comme séquence test soit la huitième globine cristallisée après 1987 (séquence extraite de l'entrée PDB 1mba (hémoglobine)), soit les mêmes six séquences prises comme exemples. Une septième séquence de Bashford (séquence extraite de l'entrée PDB 2lh4 (leghémoglobine)) a été laissée de côté dans ces premiers essais car elle pose problème même dans un alignement structural [Escalier *et al.*, 1996].

Un alignement des six séquences exemples est montré dans la figure 6.24. Celle-ci indique l'emplacement des six fragments de structure secondaire considérés comme faisant partie du noyau par Gerstein et/ou Escalier. Le même alignement est montré dans la figure 6.25 montrant cette fois l'emplacement des structures secondaires (correspondant aux 8 hélices) telles que celles-ci sont présentées dans l'article de Bashford. La figure 6.26 montre l'arrangement de ces motifs dans l'espace pour la structure de la chaîne α de l'hémoglobine humaine. Enfin, le tableau 6.6 fournit la liste des motifs structuraux trouvés par Escalier et les positions qu'ils recouvrent pour les six globines exemples. Il est important d'observer ici que, bien que chaque motif du noyau forme une suite contiguë de résidus, ce n'est pas ainsi qu'Escalier repère ces motifs. Son algorithme, appelé 'mescan', permet en fait d'identifier de vraies formes tridimensionnelles répétés parmi un ensemble de macromolécules. Ses blocs structuraux peuvent ainsi comporter des résidus non contigus sur la chaîne. Les numéros des blocs dans le tableau 6.6 ne correspondent donc pas aux blocs identifiés par mescan mais aux éléments de structures secondaires contenus dans les blocs trouvés par mescan.

Observons enfin que les blocs du noyau des globines sont plus purement structuraux que dans le cas des P450. L'hélice F semble ainsi être la seule liée à l'activité de la protéine.

6.3.4.2 Résultats

La présentation des résultats obtenus avec les globines suit celle adoptée pour les P450. Comme auparavant, nous donnons d'abord ceux produits par CoSamp, puis les résultats de CLUSTAL W et MACAW + Gibbs.

Les résultats de CoSamp représentent trois essais qui peuvent être rangés en deux groupes. Cette fois nous n'avons fait usage que de LeCombi en première phase de CoSamp. Les paramètres

Figure 6.24: Éléments de la structure secondaire des globines faisant partie du noyau. Les boîtes en gris foncé, moyen ou clair ou en blanc indiquent les blocs 3D trouvés par Escalier [Escalier *et al.*, 1996] (4 motifs repérés, d'abord le gris foncé, puis le moyen, le clair et enfin le blanc). Les traits noirs au-dessus des séquences indiquent les blocs de Gerstein.

Bloc	Hélice	Position 1mbd	Position 2lhb	Position 2hbg	Position 1ecd	Position 2hhbA	Position 2hhbB	Long. du bloc
1	A	4-17	13-26	4-17	3-16	4-17	5-18	14
2	B-C	24-43	33-52	26-45	19-38	24-43	23-42	20
3	E	58-72	70-84	55-69	55-69	55-69	60-74	15
4	F	79-91	94-106	79-98	76-88	76-88	81-93	13
5	G	100-107	115-122	102-109	96-103	97-104	102-109	8
6	H	125-140	131-146	127-142	118-133	122-137	127-142	16

Tableau 6.6: Blocs structuraux trouvés par V. Escalier [Escalier *et al.*, 1996] sur les globines.

Figure 6.25: Éléments de la structure secondaire des globines : huit hélices localisées manuellement par Bashford [Bashford *et al.*, 1987].

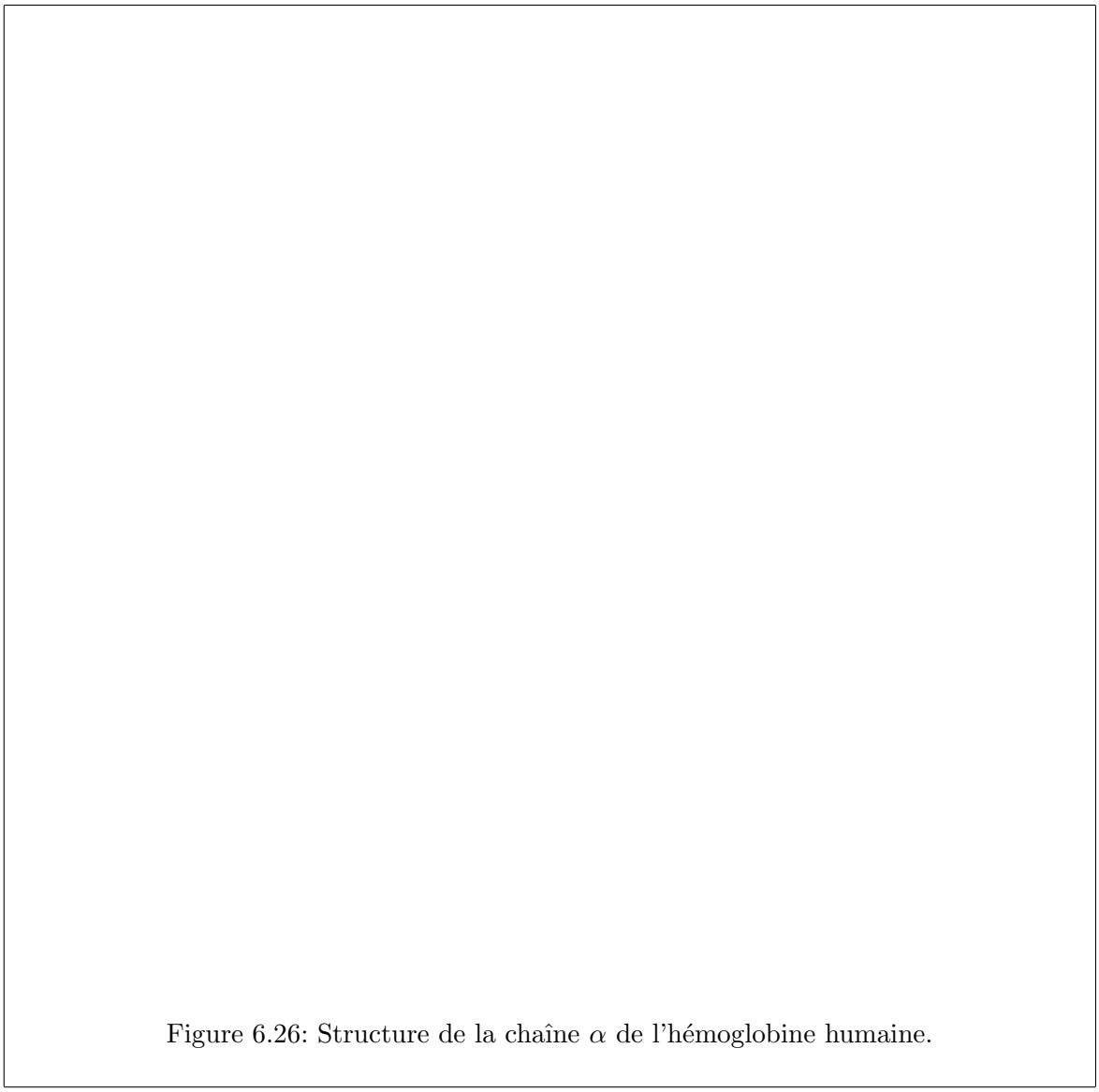


Figure 6.26: Structure de la chaîne α de l'hémoglobine humaine.

utilisés ainsi que l'indication des ensembles tests employés est donnée ci-dessous :

Groupe 1

- Séquences exemples : 2hhbA, 2hhbB, 2lhb, 1mdb, 2hgb, 1ecb
- Séquences tests :

  essai 1 1mba

  essai 2 les mêmes séquences prises comme exemples
- Algorithme utilisé en première phase : LeCombi
- Paramètres :
  - quorum : 65%
  - nombre de jokers : 8
  - nombre de substitutions autorisées : 0
  - longueur des modèles : 12
- Couverture pondérée restreinte :

$$CCPR = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in C, \\ (\Sigma, 8), \\ (S, 0)\ \textit{pour tous les autres ensembles} \end{array} \right\}$$

  avec :

$$C = \left\{ \begin{array}{ll} \{I, L, M, V\} & (\textit{hydrophobe}), \\ \{A, G\} & (\textit{très petit}), \\ \{A, S, T\} & (\textit{petit}), \\ \{C\} & (\textit{cystéine}), \\ \{F, Y, W\} & (\textit{aromatique}), \\ \{K, H, R\} & (\textit{basique}), \\ \{E, D\} & (\textit{acide}), \\ \{N, Q\} & (\textit{glutamine et glutamate}), \\ \{G, P\} & (\textit{apparaît dans les structures contraintes}) \end{array} \right\}$$

- Temps d'exécution de LeCombi : 10 secondes (sur une Sparc Station 20)
- Nombre de modèles initiaux : 850 représentant en fait 666 ensembles d'occurrences distincts

Groupe 2

- Séquences exemples : 2hhbA, 2hhbB, 2lhb, 1mdb, 2hgb, 1ecb
- Séquences tests : les mêmes séquences exemples
- Algorithme utilisé en première phase : LeCombi
- Paramètres :
  - quorum : 100%
  - nombre de jokers : 8
  - nombre de substitutions autorisées : 1

Figure 6.27: Résultat première série — premier essai globines (LeCombi, six séquences, quorum de 65%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

- longueur des modèles : 14

- Couverture pondérée restreinte : la même que pour le Groupe 2
- Temps d'exécution de LeCombi : 783 secondes (sur une Sparc Station 20)
- Nombre de modèles initiaux : 2471 représentant 1889 ensembles d'occurrences distincts

Les figures 6.27 et 6.28 correspondent au groupe 1, la figure 6.29 correspond au groupe 2.

Les figures 6.30 et 6.31 quant à elles présentent les résultats obtenus avec CLUSTAL W et la méthode d'échantillonnage de Gibbs intégrée dans MACAW. Dans le cas de ce dernier, la longueur des blocs recherchés a varié entre 4 et 12. Le jeu d'essai pour les deux algorithmes est le même (six séquences) que celui utilisé dans CoSamp.

6.3.4.3 Discussion

Les commentaires généraux que nous pouvons faire concernant les résultats observés avec CoSamp sur les globines demeurent essentiellement les mêmes que ceux relatifs aux P450.

Figure 6.28: Résultat première série — deuxième essai globines (LeCombi, six séquences, quorum de 65%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

Figure 6.29: Résultat deuxième série — globines (LeCombi, six séquences, quorum de 100%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

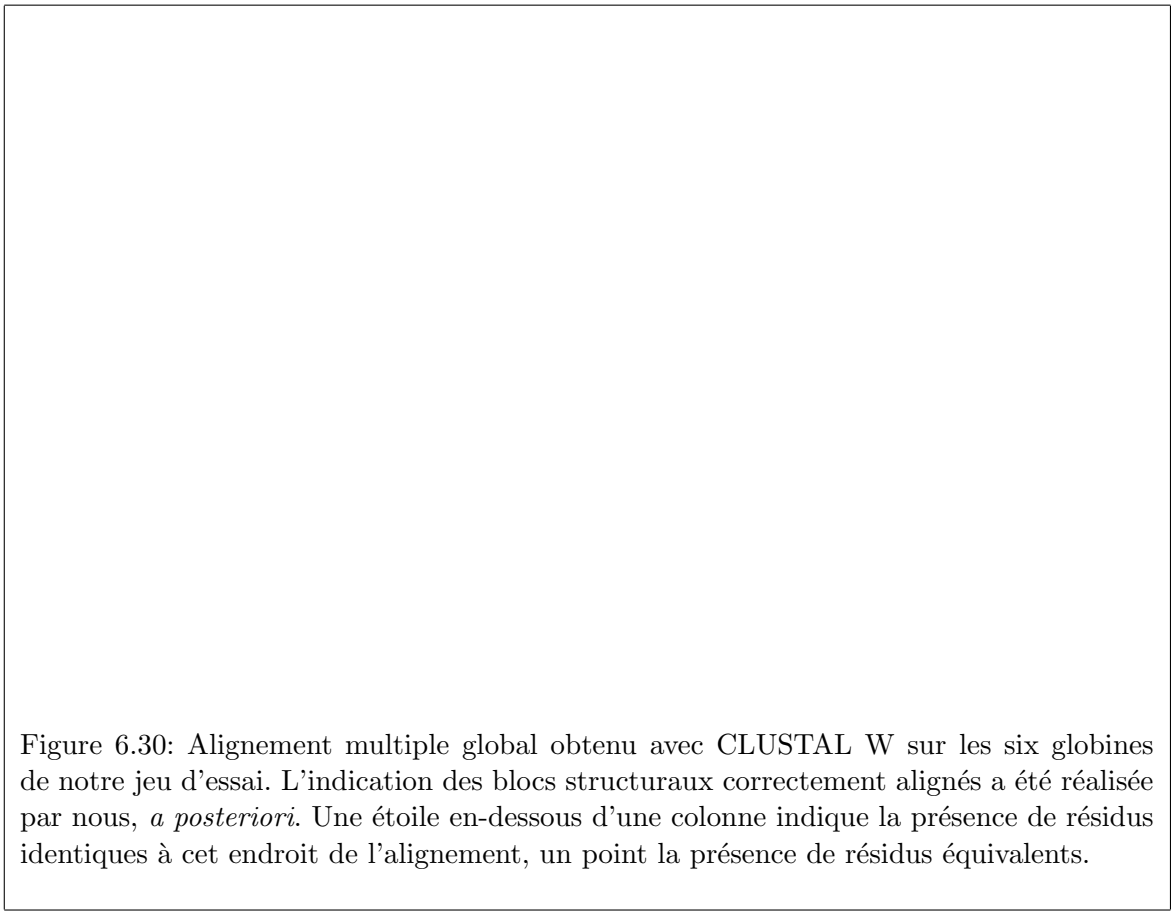


Figure 6.30: Alignement multiple global obtenu avec CLUSTAL W sur les six globines de notre jeu d'essai. L'indication des blocs structuraux correctement alignés a été réalisée par nous, *a posteriori*. Une étoile en-dessous d'une colonne indique la présence de résidus identiques à cet endroit de l'alignement, un point la présence de résidus équivalents.

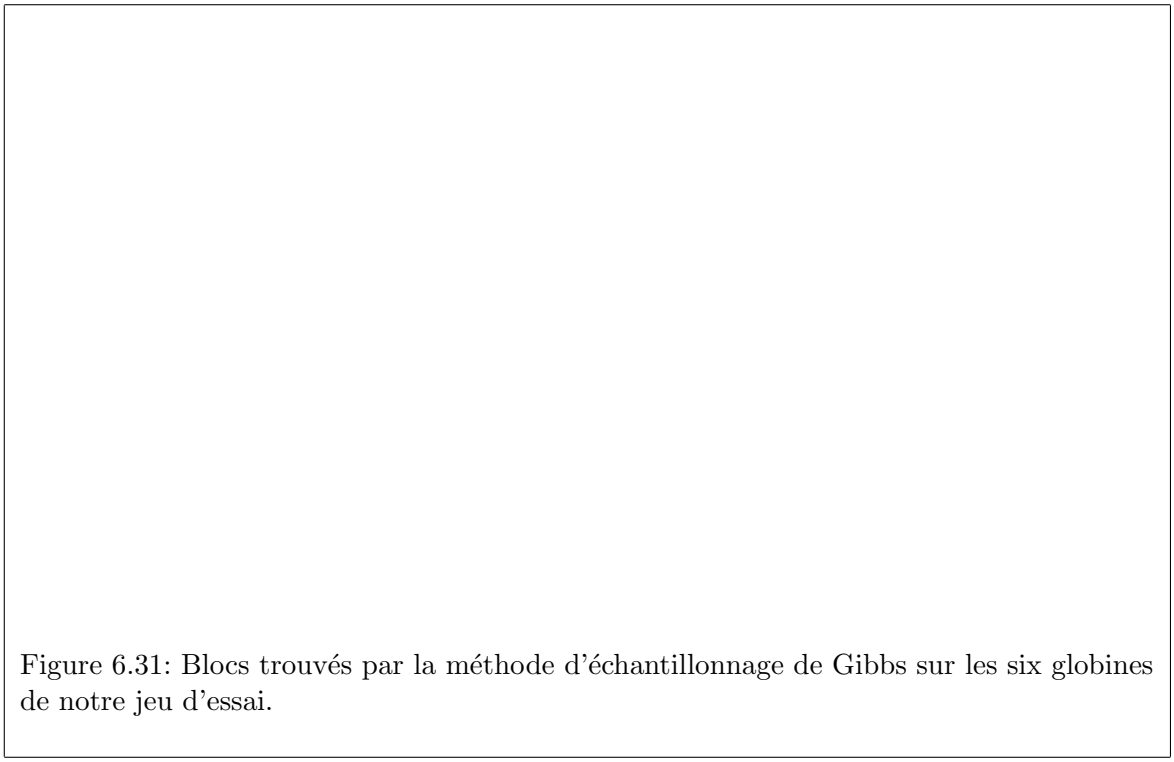


Figure 6.31: Blocs trouvés par la méthode d'échantillonnage de Gibbs sur les six globines de notre jeu d'essai.

Une observation supplémentaire importante concerne la 'meilleure qualité' des modèles trouvés par CoSamp dans ce cas. Nous proposons quelques explications à cela, mais il faut noter que celles-ci sont pour l'instant purement intuitives. La première est que les blocs structuraux à identifier en tant que blocs lexicaux sont de longueurs plus petites et surtout plus homogènes. Une seconde explication est que nous disposons dans ce cas d'un ensemble plus grand de séquences. Ceci nous permet de travailler avec un quorum relativement bas (65%, c'est-à-dire $q \geq 4$). L'algorithme de la première phase peut alors plus facilement localiser les blocs lexicaux correspondant aux blocs structuraux sur les q séquences de l'ensemble qui sont les plus proches les unes des autres. Il lui suffit ensuite de retrouver les occurrences sur les séquences manquantes, mais arrivé à ce stade, il dispose d'un alignement de q mots qui contient plus d'information que le simple nom du modèle. Cette identification plus aisée des blocs sur les séquences d'un ensemble qui sont les plus proches suggère qu'il pourrait être intéressant d'introduire dans le cadre d'une recherche de mots communs l'idée d'une comparaison ayant pour base un arbre de classification. De telles méthodes existent, surtout dans le cas global, mais ainsi que nous l'avons vu dans le chapitre 4, l'arbre utilisé est toujours, soit connu dès le départ, soit construit avant l'alignement. Signalons que dans le cas de CLUSTAL W, l'arbre construit ne réussit à regrouper que 2 des 6 séquences du jeu. La raison en est probablement que ces séquences sont trop peu conservées pour que des comparaisons deux à deux (les seules pouvant servir de base à l'établissement de l'arbre avant l'alignement multiple) puissent être vraiment utiles. Construire l'arbre et l'alignement en même temps est bien sûr un problème considérablement plus difficile, mais cet exemple est peut-être une illustration de son importance.

À nouveau ici, l'alignement global réalisé par CLUSTAL W positionne en partie correctement les six blocs considérés comme appartenant au noyau structural des globines par Es-

calier, ainsi que l'hélice D que CoSamp identifie également. Par contre aussi bien CLUSTAL W que CoSamp placent incorrectement un résidu de cette hélice : d'après Bashford, le bloc contient en effet un trou en avant-dernière position sur la séquence 2hgb, trou qu'aucun des deux algorithmes n'est capable de placer correctement (LeCombi autorise sur cet exemple une substitution mais pas de trou). D'une façon générale, l'alignement de CLUSTAL W présente plusieurs différences importantes avec l'alignement structural global réalisé par Bashford [Bashford *et al.*, 1987].

6.3.5 Les HTH ('Helix-Turn-Helix')

6.3.5.1 Introduction

L'exemple traité dans cette section est le dernier que nous présentons dans la série des protéines qui possèdent une forte conservation au niveau de leurs structures, mais très faible au niveau de leurs séquences.

Les protéines avec lesquelles nous travaillons ici appartiennent en fait à des familles souvent très diverses dont le seul point commun est leur capacité à se fixer à l'ADN (la CRP dont nous avons parlé dans la section 6.2.3 en est un exemple parmi d'autres). Pour que cette fixation puisse avoir lieu, il faut que la protéine reconnaisse un site particulier sur la macromolécule d'ADN. Rappelons que celle-ci est formée de deux brins anti-parallèles composés de bases complémentaires qui s'apparient à l'intérieur de l'hélice. La face externe de cette double hélice ne porte donc pratiquement aucune trace de l'identité des nucléotides situés à l'intérieur puisqu'elle constituée essentiellement du squelette phosphate-sucre. Pour que la protéine puisse établir un contact avec les bases nucléotidiques, il faut donc que la structure locale de l'ADN soit déformée (coude, ouverture) ou que la protéine adopte une conformation lui permettant de pénétrer à l'intérieur du grand sillon de l'ADN. Dans le cas présent, cette forme lui est conférée par un motif structural composé d'une hélice suivie par un coude puis par une autre hélice (voir la figure 6.32) et appelé un motif HTH (de l'anglais 'Helix Turn Helix').

Comme ces protéines sont d'origines diverses, elles ne présentent souvent aucune autre ressemblance structurale autre que ce motif. Bien sûr, elles ne possèdent que peu, ou pas du tout, de ressemblance en termes de séquence, sauf éventuellement au niveau du coude du motif HTH. En effet, ce dernier est en général formé d'acides aminés hydrophobes. Observons cependant que cette relative similarité ne concerne que 3 ou 4 résidus. C'est toutefois ce coude qui, la plupart du temps, permet d'identifier le motif à partir des séquences uniquement. La spécificité de la reconnaissance du site sur l'ADN par la protéine est quant à elle liée principalement à des différences situées dans les chaînes latérales du début de la seconde hélice. Cette région est appelée l'hélice de reconnaissance [Creighton, 1993]. L'autre hélice établit un contact non-spécifique avec l'ADN.

Le jeu d'essai que nous avons adopté afin de tester la capacité de nos algorithmes à repérer de faibles similarités est celui employé par Lawrence, dans son article sur la méthode d'échantillonnage de Gibbs [Lawrence *et al.*, 1993]. Ce jeu est composé de 30 séquences (dont les numéros d'accès dans la banque NFBR/PIR sont : JEBY1, DNECFS, B26499, BVECDA, A32837, C29010, A25944, A25399, QCBP2L, C55249, RPECL, RPECDU, JWEC, RGKBAP, RPECW, RPECTN, Z1BPC2, JU0083, S11945, RPECEG, RGECH, RPECCT, RPECG, RGECF, QRECC, RGECA, RGKBCP, RPECDO, RCBPL, RGBP22). La longueur du motif structural est d'environ 18 acides aminés et son emplacement sur chacune des séquences, déterminé par l'algorithme de Lawrence, est en accord avec les données expérimentales ou avec un examen direct de la structure de la protéine lorsque celle-ci a pu être cristallisée. La figure 6.33 montre un alignement des séquences à l'endroit du motif ainsi que les positions de début et de fin de

Figure 6.32: Structure hélice-coude-hélice (protéine Cro) avec son ADN.

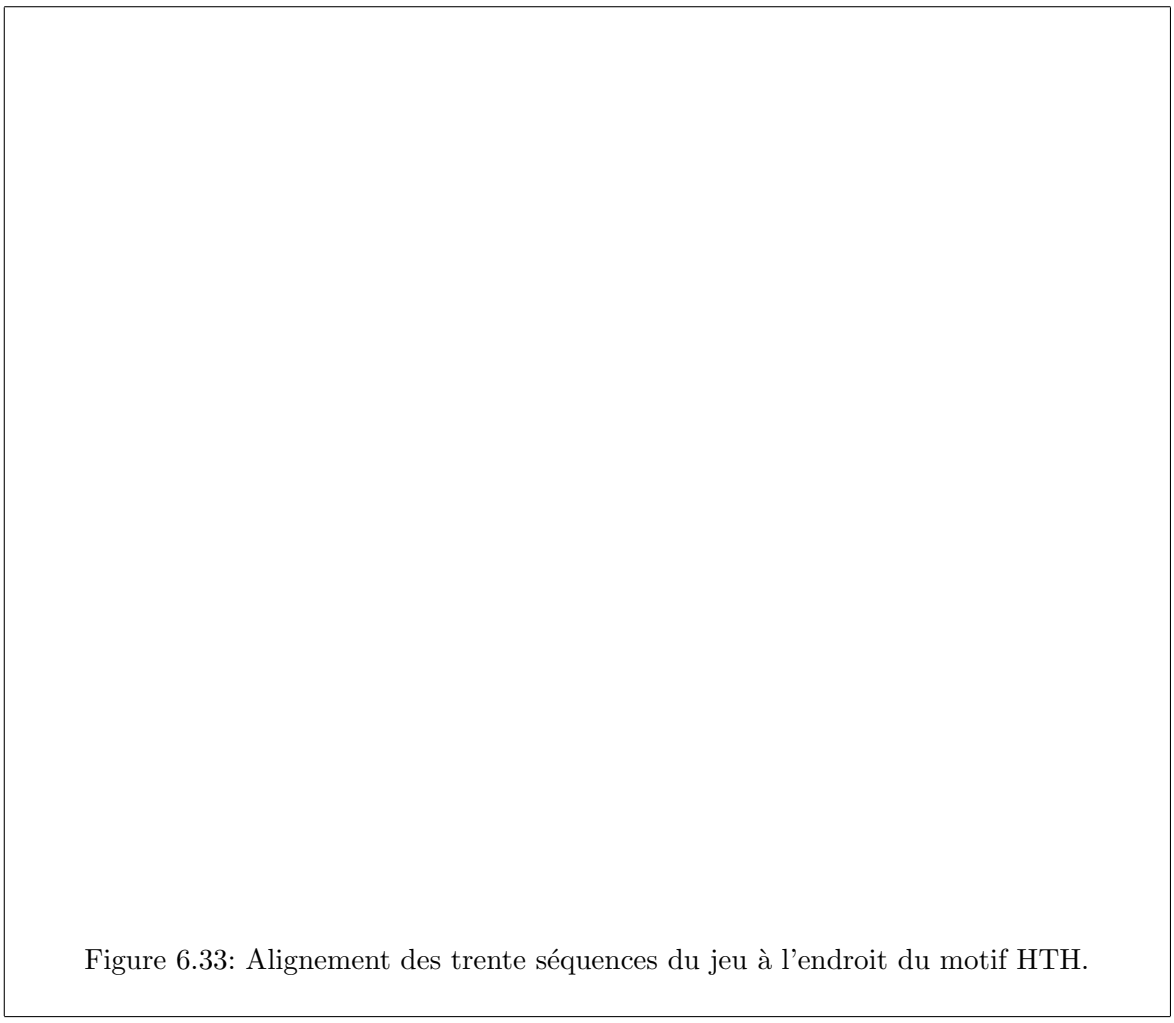


Figure 6.33: Alignement des trente séquences du jeu à l'endroit du motif HTH.

ce dernier.

Notons que Lawrence a choisi de construire un jeu d'essai duquel ont été éliminées toutes les séquences considérées trop proches les unes des autres.

Voyons maintenant les résultats obtenus avec CoSamp, ainsi qu'avec l'algorithme de Lawrence (MACAW + Gibbs) et avec CLUSTAL W.

6.3.5.2 Résultats

Comme l'ensemble des séquences du jeu d'essai est bien plus grand que dans le cas des deux illustrations précédentes, et comme l'identification du motif HTH en tant que motif lexical est connu pour être difficile, nous avons décidé initialement de fixer le quorum à une valeur très basse : 17%. Un modèle n'a donc besoin d'être présent que dans 5 parmi les 30 séquences de l'ensemble pour être valide. Nous avons également choisi de travailler avec LeCombi exclusivement. Enfin, dans un second temps, nous avons augmenté le quorum à 60%. Les résultats produits par CoSamp peuvent ainsi être rangés en deux groupes suivant la valeur attribuée à ce quorum, le premier contient deux essais et le second groupe un seul essai. Les paramètres correspondant à chacun des groupes sont les suivants :

Groupe 1

- Séquences exemples : les 30 séquences sélectionnées par Lawrence
- Séquences tests : les mêmes séquences
- Algorithme utilisé en première phase : LeCombi
- Paramètres :

  essai 1

  - quorum : 17%
  - nombre de jokers : 8
  - nombre de substitutions autorisées : 0
  - longueur des modèles : 16

  essai 2

  - quorum : 17%
  - nombre de jokers : 5
  - nombre de substitutions autorisées : 1
  - longueur des modèles : 16

- Couverture pondérée restreinte :

$$CCPR = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in C, \\ (\Sigma, 8), \\ (S, 0)\ \textit{pour tous les autres ensembles} \end{array} \right\}$$

  avec :

$$C = \left\{ \begin{array}{ll} \{I, L, M, V\} & (\textit{hydrophobe}), \\ \{A, G\} & (\textit{très petit}), \\ \{A, S, T\} & (\textit{petit}), \\ \{C\} & (\textit{cystéine}), \\ \{F, Y, W\} & (\textit{aromatique}), \\ \{K, H, R\} & (\textit{basique}), \\ \{E, D\} & (\textit{acide}), \\ \{N, Q\} & (\textit{glutamine et glutamate}), \\ \{G, P\} & (\textit{apparaît dans les structures contraintes}) \end{array} \right\}$$

- Temps d'exécution de LeCombi :

  essai 1 232 secondes (sur une Sparc Station 20)

  essai 2 11866 secondes (*idem*)

- Nombre de modèles initiaux :

  essai 1 875 représentant 77 ensembles d'occurrences distincts

  essai 2 6642 représentant 139 ensembles d'occurrences distincts

Groupe 2

- Séquences exemples : les mêmes que pour le groupe 1
- Séquences tests : les mêmes séquences
- Algorithme utilisé en première phase : LeCombi
- Paramètres :
  - quorum : 60%
  - nombre de jokers : 8
  - nombre de substitutions autorisées : 1
  - longueur des modèles : longueur maximale (égale dans ce cas à 15)
- Couverture pondérée restreinte : la même que pour le groupe 1
- Temps d'exécution de LeCombi : 10923 secondes (sur une Sparc Station 20)
- Nombre de modèles initiaux : 36

Cette fois, des résultats intéressants apparaissent dès la première phase, avant même l'étape d'évaluation.

En effet, dans le premier essai, c'est-à-dire parmi les 77 modèles associés à des ensembles d'occurrences distincts obtenus en autorisant 8 jokers, 4 seulement ne sont pas associés au motif HTH. Ces quatre modèles correspondent en réalité à un autre site présent dans un sous-ensemble des protéines du jeu : il s'agit de cinq répresseurs de *coli* (*cyt*, *ebg*, *gal*, *lac* et *pur*). 15 des 72 modèles restants possèdent une seule occurrence fausse du motif HTH. Par ailleurs, 4 des 30 séquences du jeu d'essai ne présentent jamais aucune occurrence, il s'agit de : JEBY1 ('mating hormone' - *yeast*), A25399 ('homeotic protein Antennopedia' - *fruit fly*), QCBP2L ('regulatory protein cII' -*phage lambda*) et JWEC ('DNA-invertase' - *coli*). Les résultats obtenus avec le second essai, autorisant cette fois 5 jokers, et une substitution au plus, sont sensiblement les mêmes. Enfin, avec l'essai du second groupe, lorsque le quorum est fixé à 60%, le nombre de jokers est égal à 8 et au plus une substitution est autorisée, les modèles obtenus sont tous associés au motif HTH et possèdent chacun au maximum 3 occurrences 'fausses'. Seules 3 séquences sont totalement 'manquées' (pas d'occurrences ou occurrences systématiquement fausses) : JEWC, RPECW ('*trp* repressor' - *coli*) et S11945 ('*lex*A repressor' - *coli*).

À partir des résultats de l'essai 1 du premier groupe (8 jokers, 0 erreur, $q = 17\%$), nous avons alors construit une table $\mathcal{T}$ de dimension 30×30 avec $\mathcal{T}(i, j)$ indiquant le nombre de fois où un modèle est présent à la fois sur la séquence i et la séquence j pour tout i et j. Nous pouvons y distinguer (voir la figure 6.34) trois groupes différents de séquences qui se trouvent mis ensemble. Ces groupes sont les suivants :

Ensemble 1

- *cyt* 'repressor'
- *ebg* 'repressor'
- *gal* 'repressor'
- *lac* 'repressor'
- *pur* 'repressor'
- 'regulatory protein' *fnr*

Ensemble 2

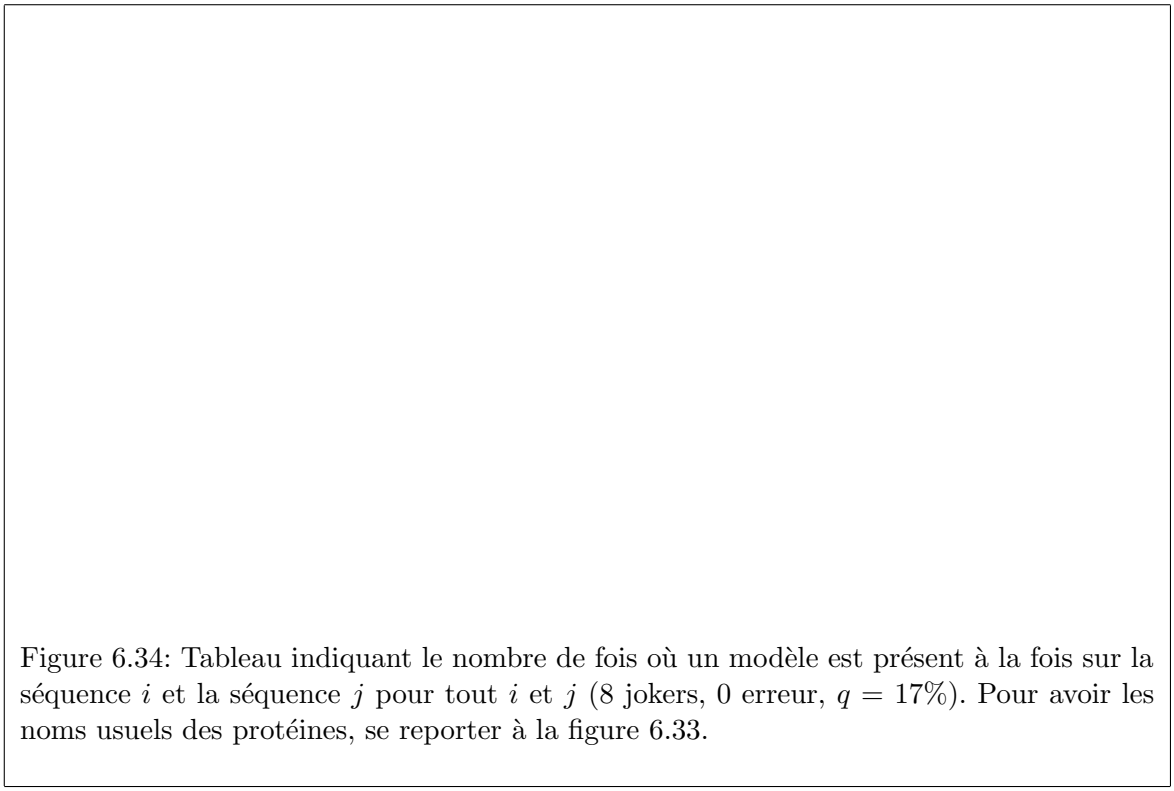


Figure 6.34: Tableau indiquant le nombre de fois où un modèle est présent à la fois sur la séquence i et la séquence j pour tout i et j (8 jokers, 0 erreur, $q = 17\%$). Pour avoir les noms usuels des protéines, se reporter à la figure 6.33.

- 'transcription initiation factor' sigmaK
- 'transcription initiation factor' sigmaB
- 'transcription initiation factor' sigmaE 'precursor'

Ensemble 3

- *dic*A 'protein'
- *deo* 'operon repressor'
- 'nitrogen assimilation regulatory protein'
- 'DNA binding protein' *fis*
- 'regulatory protein' *cro*

L'ensemble 1 regroupe ainsi tous les répresseurs du jeu d'essai sauf *trp*, *Tet*R et *deo* de *coli*, tandis que l'ensemble 2 réunit les facteurs intervenant dans l'initiation de la transcription. Enfin, l'ensemble 3 réunit des protéines apparemment complètement différentes et il serait intéressant de savoir si elles possèdent en fait un point commun autre que le motif HTH. Toutes les autres séquences du jeu semblent former des groupes isolés.

Lorsque l'étape d'évaluation est maintenant utilisée à la suite, le résultat devient encore plus intéressant et, d'une certaine façon, surprenant. Quel que soit l'essai (Groupe 1 ou 2), CoSamp obtient dans tous les cas un seul profil dont la p-value est inférieure à 10^{-5}. Ces profils sont composés à la fois d'occurrences originalement trouvées par LeCombi et de pseudo-occurrences ajoutées à l'ensemble lors de la seconde étape. Ils sont indiqués dans les figures 6.35, 6.36

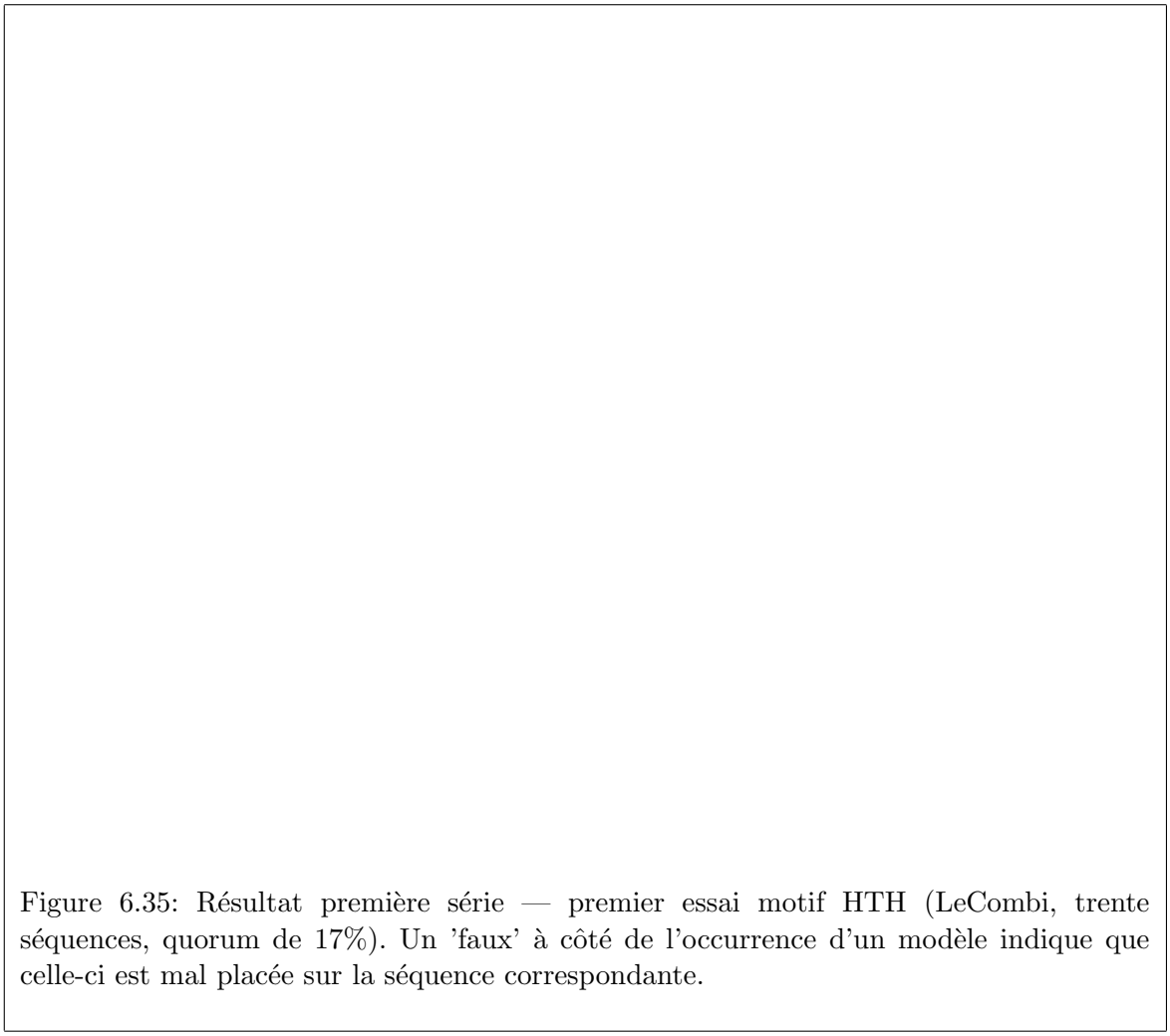


Figure 6.35: Résultat première série — premier essai motif HTH (LeCombi, trente séquences, quorum de 17%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

et 6.37. Le meilleur résultat est obtenu avec le premier essai du Groupe 1 ($q = 17\%$) où seule est fausse l'occurrence trouvée sur la séquence de numéro d'accès C29010 dans PIR ('mercuric resistance operon regulatory *mer*D protein' - *Serratia marcescens plasmid*). Observons que les séquences sur lesquelles le motif n'a pas été identifié dans la première étape apparaissent ici avec des occurrences correctes. Dans le groupe 1, le profil couvre sept résidus de la première hélice, trois résidus du coude et les six premiers résidus de l'hélice de reconnaissance tandis que dans le groupe 2, les occurrences possèdent un résidu de moins.

Enfin, la figure 6.38 présente le résultat obtenu avec l'échantillonnage de Gibbs (observons que la recherche a été faite en fixant la longueur de la matrice à 18 comme dans l'article original de Lawrence). Celui de CLUSTAL W n'est pas montré ici — il est discuté dans la section suivante.

6.3.5.3 Discussion

Notre première observation est que l'utilisation de la seconde étape dans cette illustration est, d'une certaine façon, contraire au principe de CoSamp. En effet, celui-ci est constitué d'une phase d'exploration combinatoire que l'on souhaite aussi large que possible suivie par une phase

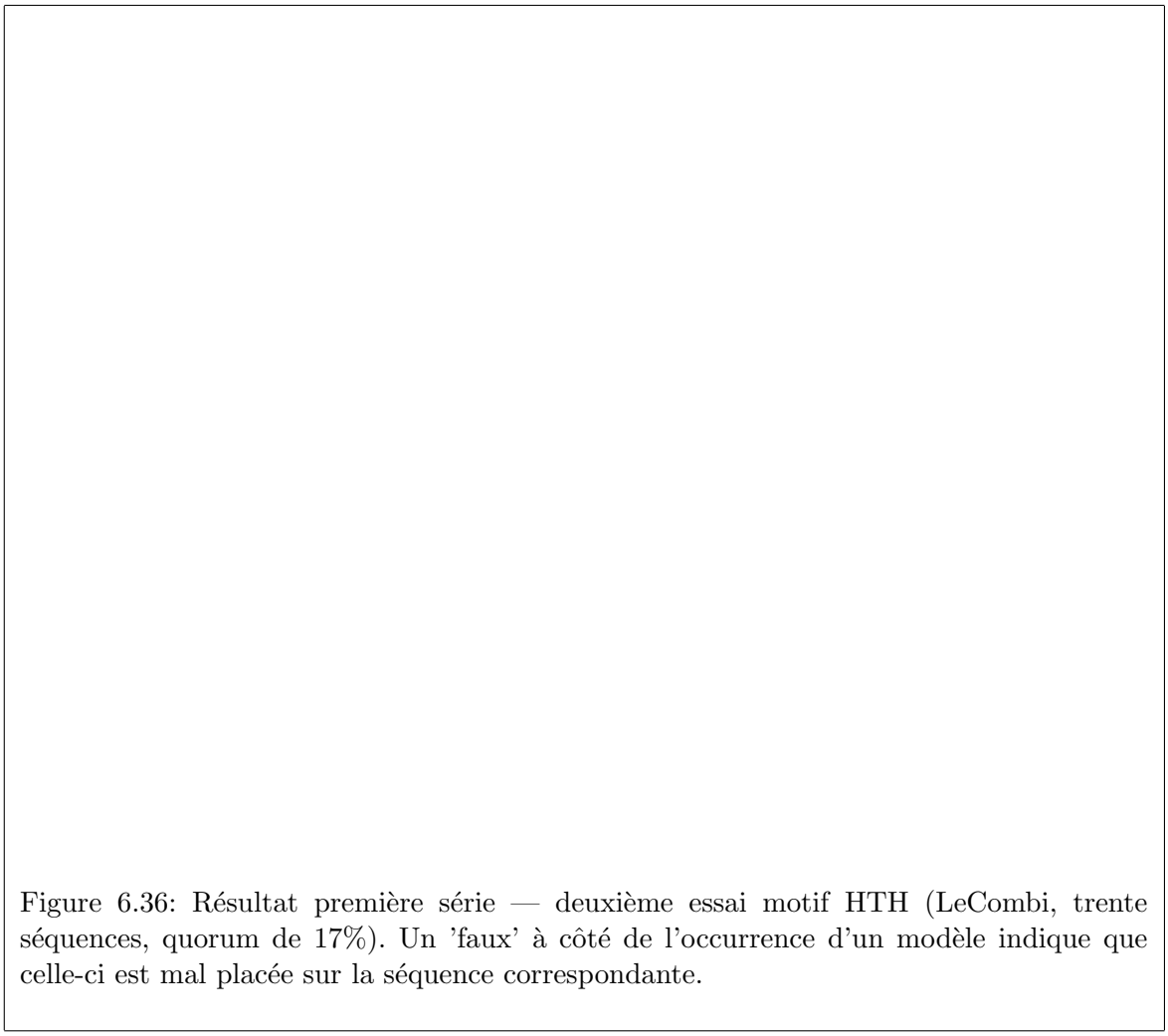


Figure 6.36: Résultat première série — deuxième essai motif HTH (LeCombi, trente séquences, quorum de 17%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

Figure 6.37: Résultat deuxième série — motif HTH (LeCombi, trente séquences, quorum de 60%). Un 'faux' à côté de l'occurrence d'un modèle indique que celle-ci est mal placée sur la séquence correspondante.

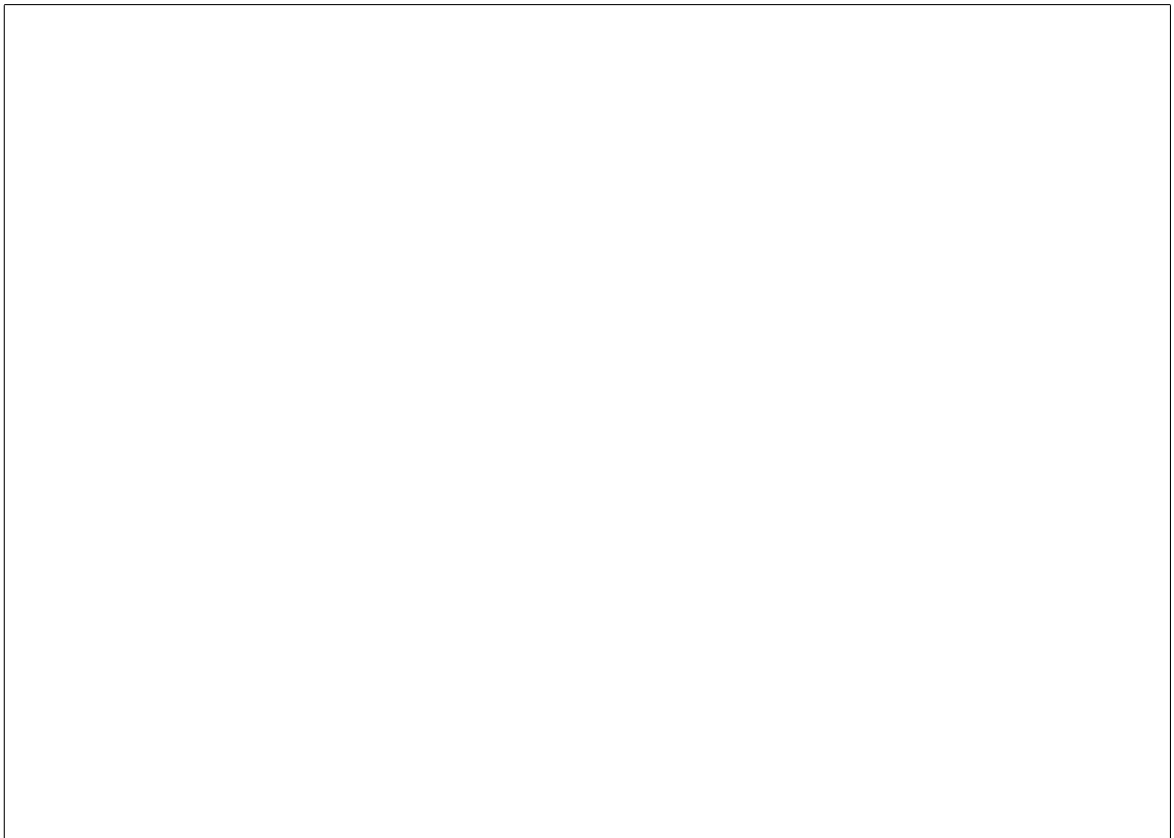


Figure 6.38: Blocs trouvés par la méthode d'échantillonnage de Gibbs sur les trente séquences de notre jeu d'essai.

d'évaluation et de classement des modèles trouvés dans la première étape. Si le quorum n'est pas égal à 100%, les ensembles d'occurrences des modèles doivent être complétés par des pseudo-occurrences mais le 'noyau' principal du modèle final reste composé des occurrences originales. Cependant, lorsque le quorum est fixé à une valeur très basse (17% dans les deux premiers exemples ici), les modèles finaux sont constitués essentiellement de telles pseudo-occurrences ce qui semble contradictoire avec la 'philosophie' originale de l'algorithme.

Par ailleurs, cet exemple suggère que d'autres approches que l'évaluation sont possibles pour la seconde étape de l'algorithme. Nous pourrions ainsi envisager de regrouper deux modèles lorsqu'ils possèdent au moins un certain nombre d'occurrences qui se chevauchent du même nombre de positions. Un tel regroupement est réalisé par Neuwald [Neuwald and Green, 1994] pendant la recherche des motifs lexicaux. Il est intéressant de noter que Neuwald n'utilise dans son illustration qu'un sous-ensemble de 15 parmi les 30 séquences du jeu de Lawrence. Il justifie cette réduction par le désir d'éliminer de l'ensemble de Lawrence les séquences considérées comme étant encore trop proches.

Le résultat produit par l'échantillonnage de Gibbs est légèrement meilleur que celui fourni par CoSamp puisque dans ce cas le motif est correctement placé sur toutes les séquences.

Avec CLUSTAL W par contre, ce que nous obtenons est cette fois complètement faux dans la mesure où l'alignement global réalisé par l'algorithme ne réussit à mettre le site correctement en correspondance que sur 6 parmi les 30 séquences. Sur ces six séquences-là, le motif est situé sur la partie C terminale des protéines ce qui explique probablement pourquoi CLUSTAL les aligne correctement. Cet exemple illustre donc un point faible de l'approche défendue par Posfai, qui consiste à établir d'abord un alignement global afin de rechercher les blocs conservés localement. Un tel alignement permet la localisation plus précise des parties conservées d'un ensemble de protéines dans la mesure où celles-ci sont relativement fixées les unes par rapport aux autres et uniformément distribuées sur les séquences. Cette approche n'est donc viable que si les séquences sont globalement similaires. Observons que cela n'est pas le cas ici, et n'est pas le cas non plus de la partie N terminale des P450 qui est probablement spécifique à chaque sous-famille et dans laquelle CLUSTAL ne parvient pas à identifier quoi que ce soit.

6.3.6 Nitrogénases et Hydrogénases

6.3.6.1 Introduction

Notre dernière illustration sera plus succinte. Elle est également plus fragile du point de vue biologique car nous ne pouvons valider ce que nous observons avec nos algorithmes avec aucun résultat antérieur. Nous ne sommes pas capables non plus, faute d'avoir des connaissances plus approfondies des protéines impliquées, de développer notre analyse. Cette illustration nous a semblé néanmoins importante parce qu'elle soulève le problème des trous.

En effet, un des partis le plus fréquemment pris est celui de considèrer qu'un bloc conservé associé à un signal ne peut présenter de trou. Or le résultat présenté ici concerne justement un motif indiqué dans PROSITE [Bairoch, 1992] comme étant spécifique d'un des sites actifs d'une famille de protéines, que LePoivre repère dans une famille différente de la première avec une délétion lorsque les modèles communs à ces deux familles sont recherchés. Plus que l'importance de ce motif en lui-même, nous voulons donc nous servir de cette illustration pour mettre en avant la nécessité de permettre la présence de trous dans la recherche de motifs. La raison pour laquelle les trous ne sont généralement pas autorisés dans les blocs est essentiellement technique : autoriser des trous augmente le temps d'exécution des algorithmes. Avec les algorithmes que nous avons développés ici, ce temps demeure cependant raisonnable dans la plupart des cas, ainsi que nous l'avons montré dans les sections de performance du chapitre 5.

Les deux ensembles de protéines dont il va être question dans cette dernière illustration font partie des protéines de la famille des nitrogénases et des hydrogénases.

Les nitrogénases sont des protéines enzymatiques responsables entre autres de la fixation de l'azote. Elles catalysent les réactions de réductions :

$$N_2 + 6H^+ \longrightarrow 2NH_3$$

ou :

$$N_2 + 8H^+ \longrightarrow 2NH_3 + H_2.$$

Cette seconde formule permet de voir que la réduction s'accompagne d'une libération d'hydrogène qui peut être couplée à une oxydation par les hydrogénases. Les nitrogénases sont des protéines dites 'fer-soufre', c'est-à-dire qu'elles contiennent toutes différents assemblages (en anglais 'cluster') constitués d'atomes de fer et de soufre, nécessaires à leur fonction. De plus, elles ont besoin d'un autre ion métallique (qui vient se fixer sur l'assemblage fer-soufre). Il s'agit, soit du molybdène (sous-familles des nitrogénases de noms mnénomiques nifK et nifD), soit, plus rarement, du vanadium (sous-familles des nitrogénases de noms mnénomiques vnfD, vnfK et vnfG). Certaines nitrogénases paraissent cependant ne fixer que le fer; elles ont pour noms anfD, anfK et anfG. Dans le cas des hydrogénases, c'est le nickel qui est nécessaire à la réaction d'oxydation. Il semble enfin que dans les deux cas, d'autres protéines interviennent dans les réactions de réduction et d'oxydation dont le rôle serait d'apporter ces métaux aux nitrogénases et aux hydrogénases [Fay, 1992] [Howard and Rees, 1994] [Robson and Postgate, 1980]. Ce sont ces protéines qui vont nous concerner ici.

Les premières, celles qui fournissent le molybdène aux nitrogénases, leur sont évolutivement liées. Elles ont pour nom mnénomique nifE et nifN. Les protéines de noms hypA, hypB, hypC, hypD, hypE et hypF, apparemment responsables quant à elles de l'apport du nickel aux hydrogénases, sont également impliquées dans la régulation de leur expression. Les deux sous-familles de ces protéines qui vont particulièrement nous intéresser sont les nifE et les hypD. Elles forment un jeu de 5 + 6 séquences dont les noms et numéros d'accès dans SWISS PROT [Bairoch and Boeckmann, 1991] sont respectivement : NIFE_AZOVI, NIFE_BRAJA, NIFE_CLOPA, NIFE_KLEPN, NIFE_RHOCA, HYPD_ALCEU, HYPD_AZOVI, HYPD_BRAJA, HYPD_ECOLI, HYPD_RHILV et HYPD_RHOCA.

6.3.6.2 Résultats

Les deux familles (nifE et hypD) ont été initialement regroupées un peu par hasard. Nous travaillions alors sur les nifE seules, et cherchions des occurrences 'avec trou' d'un des motifs fonctionnels (telles que celles observées sur les hypD). Dans les résultats présentés ici, nous avions recherché les modèles les plus longs présents dans au moins 9 des 11 séquences du jeu d'essai (fixant ainsi le quorum à 80%). Enfin, nous avions utilisé LePoivre avec la couverture suivante :

$$C = \left\{ \begin{array}{ll} \{I, L, M, V\} & (\textit{hydrophobe}), \\ \{A, G\} & (\textit{très petit}), \\ \{A, S, T\} & (\textit{petit}), \\ \{C\} & (\textit{cystéine}), \\ \{F, Y, W\} & (\textit{aromatique}), \\ \{K, H, R\} & (\textit{basique}), \\ \{E, D\} & (\textit{acide}), \\ \{N, Q\} & (\textit{glutamine et glutamate}), \\ \{G, P\} & (\textit{apparaît dans les structures contraintes}) \end{array} \right\}.$$

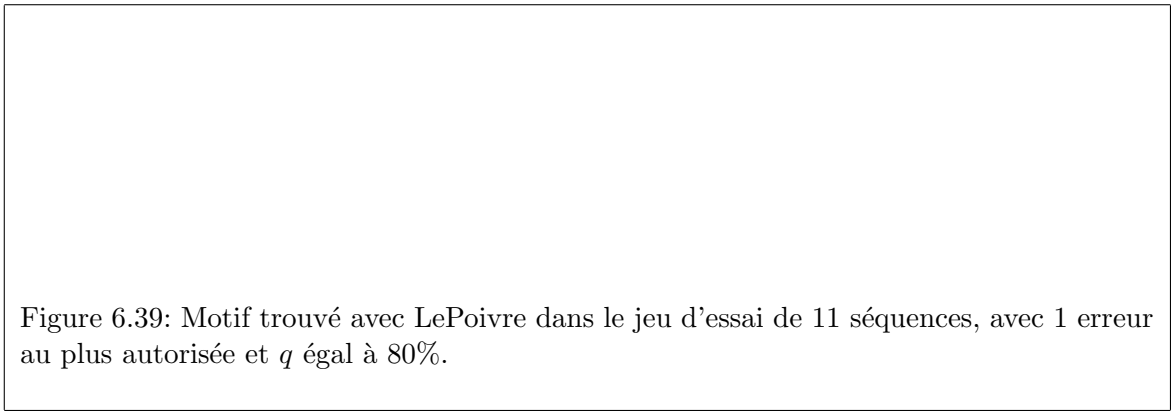


Figure 6.39: Motif trouvé avec LePoivre dans le jeu d'essai de 11 séquences, avec 1 erreur au plus autorisée et q égal à 80%.

Nous avions alors obtenu 11 modèles de longueur 9 représentant en fait le même ensemble d'occurrences. Cet unique bloc est donné dans la figure 6.39 avec les noms des séquences où il est présent ainsi que les positions. Observons que ce bloc est absent des séquences HYPD_ECOLI et HYPD_RHOCA. Il est important de noter que ce modèle est inclus dans un des deux motifs indiqués dans PROSITE comme étant associés à un site actif de la famille des nitrogénases (le motif de PROSITE est le suivant : [LIVMFYH][LIVMFST]H[AG][AGSP][LIVMQA][AG]C). La question que nous nous sommes alors posée était : est-ce que cette association est fortuite ou constitue-t-elle une information concernant les protéines de la sous-famille des nifE et des hypD, et éventuellement, à travers eux, celle des nitrogénases et des hydrogénases?

Pour essayer de le déterminer, nous avons commencé par résumer ces modèles sous la forme d'un unique motif $[KHR][ILMV][ILMV]H[AG]P[ILMV][AG]C$ que nous avons ensuite recherché dans toute la banque SWISS PROT (version 31) en autorisant une erreur (substitution ou trou) avec une version simplifiée d'Agrep [Wu and Manber, 1992a] [Wu and Manber, 1992b] que nous avions implémentée. Nous avons alors découvert que celui-ci est présent également dans 12 nifD, 1 nitrogénase fixant le fer (anfD), 2 fixant le vanadium (vnfD) d'espèces différentes, ainsi que dans une protéine hypothétique de *coli* complètement différente (YJGB_ECOLI). La liste de ces séquences est donnée dans la figure 6.40.

Notons que si nous rajoutons la contrainte que les 3 positions strictement conservées dans le bloc initial (celles marquées d'une étoile dans la figure 6.39) ne doivent pas tolérer d'erreur, alors la recherche dans la banque fournit le même résultat à l'exception de la protéine hypothétique de *coli*, dont nous pouvons supposer qu'elle constituait un faux-positif.

Nous avons ensuite poursuivi notre étude de ce même jeu de six séquences initial en recherchant la présence éventuelle de mots communs avec LeCombi. Nous avons donc cette fois permis des jokers en plus d'erreurs (une substitution au plus dans ce cas). La couverture pondérée restreinte avec laquelle nous avons travaillé est :

$$CCPR = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in C, \\ (\Sigma, 5), \\ (S, 0)\ \textit{pour tous les autres ensembles} \end{array} \right\}$$

Figure 6.40: Séquences de la banque où le motif de la figure 6.39 est trouvé en permettant une erreur. Ce motif n'est plus retrouvé sur la séquence marquée d'un '-' lorsque l'on ne tolère pas d'erreur à certaines positions du motif (celles indiquées par une étoile dans la figure 6.39).

Figure 6.41: Motif trouvé avec LeCombi dans le jeu d'essai de 11 séquences, avec 5 jokers et 1 substitution au plus autorisée et q égal à 100%.

avec :

$$C = \left\{ \begin{array}{ll} \{I, L, M, V\} & (\textit{hydrophobe}), \\ \{A, G\} & (\textit{très petit}), \\ \{A, S, T\} & (\textit{petit}), \\ \{C\} & (\textit{cystéine}), \\ \{F, Y, W\} & (\textit{aromatique}), \\ \{K, H, R\} & (\textit{basique}), \\ \{E, D\} & (\textit{acide}), \\ \{N, Q\} & (\textit{glutamine et glutamate}), \\ \{G, P\} & (\textit{apparaît dans les structures contraintes}) \end{array} \right\}.$$

Nous avons, par ailleurs, fixé le quorum à 100%. Huit modèles de longueur maximale (13) ont alors été identifiés, représentant 4 ensembles d'occurrences distincts, dont 2 sont en fait identiques à un décalage de positions près. Parmi les 3 ensembles restants, 2 mettent en correspondance des fragments des deux familles de protéines situés à des extrémités opposées des macromolécules et nous avons donc décidé de les ignorer pour l'instant. Par contre, le troisième ensemble nous a paru intéressant (il est donné dans la figure 6.41). On peut le résumer sous la forme du motif $[FY]GXPXXX[AG]XFX[GP][ILVM]$ où X dénote n'importe quel acide aminé. Observons que le G en troisième position représente l'intersection des deux ensembles de la couverture, $\{A, G\}$ et $\{G, P\}$, qui sont rencontrés dans les modèles à cet endroit-là). Nous avons recherché ce motif dans la banque avec une substitution au plus (pas de trous cette fois puisque ceux-ci n'étaient pas permis dans LeCombi). Le résultat de cette recherche est donné dans la figure 6.42.

En fixant comme précédemment les positions où un symbole unique apparaît dans les occurrences de l'ensemble de départ (celles marqués d'une étoile dans la figure 6.41), c'est-à-dire, en interdisant des substitutions en ces positions, le résultat se réduit alors aux séquences indiquées par un '+' dans la figure 6.42.

Nous pouvons observer que ce second motif semble être moins sélectif des deux familles de protéines. Par contre, il est présent dans toutes les séquences de notre jeu initial, y compris HYPD_ECOLI et HYPD_RHOCA d'où le premier motif était, lui, absent.

Nous avons alors voulu déterminer dans quelle mesure ces deux motifs considérés ensemble permettent de sélectionner les protéines qui nous intéressent (les nifE, nifN et hypD ainsi que

Figure 6.42: Séquences de la banque où le motif de la figure 6.41 est trouvé en permettant une substitution. Ce motif n'est plus retrouvé sur les séquences marquées d'un '-' lorsque l'on ne tolère pas de substitution à certaines positions du motif (celles indiquées par une étoile dans la figure 6.41) — Continue sur la page suivante.

Figure 6.42: Séquences de la banque où le motif de la figure 6.41 est trouvé en permettant une substitution. Ce motif n'est plus retrouvé sur les séquences marquées d'un '-' lorsque l'on ne tolère pas de substitution à certaines positions du motif (celles indiquées par une étoile dans la figure 6.41).

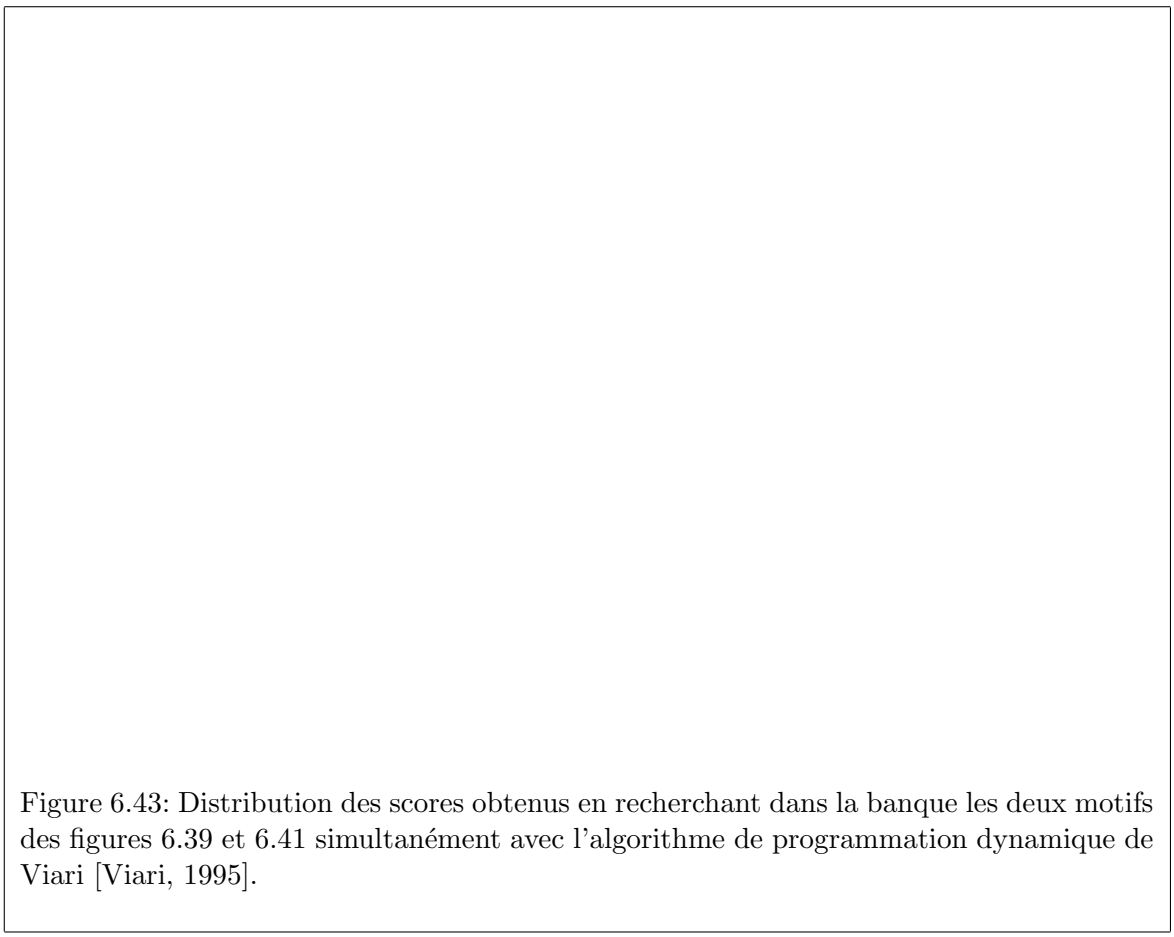


Figure 6.43: Distribution des scores obtenus en recherchant dans la banque les deux motifs des figures 6.39 et 6.41 simultanément avec l'algorithme de programmation dynamique de Viari [Viari, 1995].

les nitrogénases liées évolutivement aux nifE et nifN) et celles-là seulement.

Pour cela, il nous fallait un algorithme qui nous permette de rechercher ces motifs de manière plus souple que ne le fait Agrep, c'est-à-dire, un algorithme qui nous fournisse un score et non plus seulement une indication de la présence ou non de ces motifs parmi les séquences de la banque. Nous avons alors commencé par établir, pour chacun des motifs, un profil comme celui que nous utilisons dans CoSamp d'après la technique de Waterman [Goldstein and Waterman, 1994]. Puis nous avons calculé le score maximal de ces profils sur chacune des séquences. Deux différences fondamentales ont cependant été introduites à ce stade. La première est que les deux profils correspondant aux deux motifs sont recherchés simultanément et dans l'ordre : le premier motif doit se localiser avant le second puisque c'est ainsi qu'ils apparaissent dans les séquences du jeu d'essai. D'autre part, comme nous voulons cette fois permettre des trous (à cause du premier motif), nous utilisons un algorithme de programmation dynamique pour obtenir ce meilleur placement des deux motifs [Viari, 1995]. Observons que le poids attribué à un trou dans un motif est égal à la plus forte valeur observée dans le profil. La figure 6.43 donne la distribution des scores qui sont obtenus sur SWISS PROT (version 33).

La queue de cette distribution, c'est-à-dire, les séquences ayant un score supérieur à 47, sont indiquées dans la figure 6.44. La séquence de plus fort score est une protéine du même type que les hypD, mais présente dans un autre organisme (les hypD sont trouvés dans les bactéries

comme *Escherichia coli* et les hupD dans des organismes de plantes). Parmi les séquences restantes, apparaissent toutes les hypD, toutes les nifE, 13 parmi les 17 nifD, 6 nitrogénases (1 nifK, 2 nifN, 2 vndF et 1 anfD) et une seule protéine hypothétique et de fonction inconnue.

6.3.6.3 Discussion

Une étude beaucoup plus approfondie serait nécessaire pour savoir si ces premiers résultats présentent un réel intérêt biologique. Ainsi que nous l'avons dit, la principale difficulté rencontrée dans cette analyse est que nous ignorons à près tout de ces protéines, qui semblent par ailleurs exercer une activité très complexe dont nous n'avons fait ici que mentionner l'aspect le plus connu. S'il s'avèrait que ce que nous observons est indicatif d'une vraie homologie de fonction entre les hypD et les nifE et nifD (ainsi que les nitrogénases en général), cela renforcerait bien sûr notre conviction de l'importance des trous. Il serait alors intéressant de voir de façon plus générale dans quelle mesure les trous ne sont pas beaucoup plus courants dans les blocs de similarité que l'on peut associer à l'activité d'une même famille de protéines, ou de familles évolutivement distantes.

Figure 6.44: Séquences de la banque où les deux motifs des figures 6.39 et 6.41 sont trouvés simultanément par programmation dynamique [Viari, 1995] avec un score supérieur ou égal à 47. Les séquences sont classées par ordre décroissant des scores.

Chapitre 7

Critiques et perspectives

We play at Paste -
Till qualified, for Pearl -
Then, drop the Paste -
And deem ourself a fool -

The Shapes - though - were similar -
And our new Hands - Learned Gem-*Tactics -*
Practicing Sands -

Poème #320. Emily Dickinson.
The Complete Poems of Emily Dickinson.
Thomas H. Johnson (Ed.). Faber and Faber, 1975.

Ce travail a soulevé plus de problèmes qu'il n'a réussi à en résoudre. Certains ont été mentionnés auparavant, nous souhaitons dans ce dernier chapitre les reprendre de manière plus systématique. L'examen critique de notre approche, tant au niveau informatique qu'à celui de la biologie, nous conduira parfois à remettre en cause un certain nombre de concepts, comme celui d'une comparaison multiple. Nous proposerons également des développements possibles des définitions de ressemblance et des algorithmes introduits précédemment.

Ce chapitre est divisé en deux parties. La première considère les aspects plutôt algorithmiques et les suggestions qui y sont faites pour l'avenir sont donc le plus souvent des extensions immédiates de ce travail. La seconde partie traite les aspects plus étroitement liés à la biologie et les perspectives correspondantes exigent une réflexion plus approfondie de la notion de ressemblance. Bien sûr, la frontière entre ces deux parties peut être moins nette que nous ne l'affirmons ici.

7.1 Les critiques et perspectives portant sur l'aspect algorithmique

Les critiques présentées dans cette section font exclusivement référence à l'approche lexicale que nous avons adoptée. Notons toutefois que les problèmes rencontrés, et qui ont été principalement indiqués dans le chapitre consacré aux illustrations, s'appliquent pour la plupart à d'autres types d'approches également — par exemple, celles purement structurales. Les développements que ces critiques suggèrent sont le plus souvent de même nature.

Les difficultés algorithmiques que nous avons rencontrées sont essentiellement liées à la complexité (due en particulier à la méthode de construction par récurrence des modèles) et à la forme parfois insatisfaisante des résultats produits.

En ce qui concerne la complexité algorithmique, celle obtenue par nos algorithmes est intrinsèquement attachée au problème que nous essayons de résoudre. Cela est principalement le cas en ce qui concerne le nombre de modèles (et de leurs occurrences) que ces algorithmes doivent manipuler pour parvenir aux résultats demandés. Ainsi, les facteurs les plus importants dans ce nombre et, par conséquent, ceux ayant la plus forte influence sur le temps d'exécution, interviennent sous la forme d'un terme exponentiel. Il s'agit du nombre e d'erreurs autorisées (e.g. k^e où k est la longueur des modèles) et/ou de cette longueur k lorsque l'alphabet sur lequel les modèles sont définis est dégénéré (e.g. g^k où g mesure le degré de non-transitivité d'une couverture). Or le poids de ces facteurs ne peut être diminué lorsque nous recherchons toutes les solutions possibles du problème initial (c'est-à-dire, tous les modèles). Par contre, ces facteurs n'interviennent en pratique que lorsque la longueur des modèles est encore relativement petite (autour de 4-6). Ils cessent d'avoir une influence aussi importante dès que la longueur des modèles qui demeurent valides devient grande, car ceux-ci sont alors peu nombreux. Modifier le mode de construction des modèles, par exemple en engendrant directement tous ceux suffisamment longs, n'apporte pas une solution au problème. Il y a essentiellement deux raisons à cela. La première est qu'il est difficile de déterminer à partir de quelle longueur il n'y aura plus qu'un nombre réduit de modèles encore valides. La seconde est qu'en général cette longueur est trop grande pour qu'une génération automatique de tous les modèles possibles puisse se faire en un temps raisonnable. L'algorithme de Lawrence (section 4.2.2.2), qui produit les résultats les plus comparables aux notres, résoud le problème, c'est-à-dire, il évite d'avoir à passer par le 'pic' des petits modèles (il s'agit pour lui de matrices), essentiellement parce qu'il n'est pas exhaustif. Il n'explore ainsi qu'une partie de l'espace de toutes les solutions possibles compatibles avec la définition de ressemblance qu'il établi. Dans la plupart des cas, la partie explorée est la 'bonne', c'est-à-dire, la matrice obtenue à la fin est bien celle qui maximize la quantité observée, mais ceci n'est pas garanti absolument. Cependant nous avons vu dans la section 4.2.3.3 qu'il existe des techniques permettant de trouver tous les mots identiques communs à un ensemble de chaînes qui, par l'utilisation qu'ils font d'une structure de données spéciale (arbre des suffixes ou DAWG), sont plus efficaces que les notres en l'absence d'erreurs ou de dégénérescence de l'alphabet. La complexité obtenue est ainsi en $O(n.N)$ alors que la notre est en $O(n.N.\log k)$ ou en $O(n.N.k)$ où n est la longueur moyenne des chaînes et N leur nombre. Cela nous amène à poser un certain nombre de questions qui sont :

- est-il possible d'attribuer des noms aux mots répétés de longueur k ou k_{max} de N chaînes en $O(n.N)$ temps (ou en moins de $O(n.N.\log k)$ opérations) sans avoir besoin de construire une structure élaborée comme un arbre des suffixes (ni utiliser une table indexée ou de hachage), c'est-à-dire donc, en procédant 'à la volée'?

- pouvons-nous envisager la construction d'un arbre/automate des facteurs/suffixes qui tienne compte d'une relation de similarité non-transitive entre mots d'une ou de plusieurs chaînes? Est-ce que l'utilisation d'une telle structure nous permettrait d'éliminer le terme en k ou $\log k$ comme pour les relations d'identité? Est-ce que toute construction utile sera nécessairement trop coûteuse en mémoire?

- d'une manière plus simple, est-ce qu'un arbre/automate des facteurs/suffixes exacts pourrait être utilisé uniquement pour faciliter le parcours dans l'arbre des modèles, et non pour stocker ces derniers, cela quelle que soit la définition de ressemblance adoptée?

Ces questions demeurent ouvertes pour l'instant.

Il paraît assez clair que, lorsqu'une contrainte de quorum q strictement supérieure à 1 est appliquée, des économies plus grandes que celles que nous avons déjà réalisées devraient être possibles. Il suffit pour cela d'observer que lorsqu'un certain modèle m (un modèle comme un mot) ou M (un modèle comme un produit cartésien d'ensembles d'une couverture) ne vérifie plus cette contrainte, alors ce n'est pas seulement tous les modèles m'/M' ayant m/M comme préfixe dont nous pouvons affirmer qu'ils ne vérifient pas non plus cette contrainte, mais tous les modèles m'/M' ayant m/M comme sous-modèle! Or cette observation n'est jamais prise en considération dans aucun de nos algorithmes, simplement parce que l'arbre des modèles est parcouru en profondeur et qu'il n'est jamais effectivement construit. Le faire exigerait bien sûr plus de mémoire, la question est alors, y a-t-il un équilibre à trouver entre mémoire occupée et temps de calcul?

En ce qui concerne les résultats obtenus, les problèmes rencontrés ont été de natures diverses. Le premier est que ces résultats représentent souvent une quantité d'information encore assez considérable qu'il faut essayer de condenser d'une certaine façon. La 'condensation' des modèles peut se faire de deux manières principales :

- les modèles peuvent être regroupés en 'méta-modèles verticaux' : c'est ce que nous faisons déjà d'une première manière simple lorsque nous réunissons en un seul deux modèles ayant exactement les mêmes ensembles d'occurrences, ou éliminons les modèles strictement inclus dans un autre en termes de ces ensembles;

- les modèles peuvent être réunis en 'méta-modèles horizontaux', c'est-à-dire en trains de modèles dont les occurrences se succèdent dans le même ordre sur chacune des chaînes de l'ensemble que nous comparons.

Cette seconde manière impose l'établissement d'un ordre (partiel) de succession des modèles le long des chaînes. Elle est particulièrement intéressante lorsque ce qui est recherché représente une succession de répétitions contiguës directes ou miroir, un palindrome flou, ou encore un signal constitué par une suite de signaux séparés entre eux d'une certaine distance. Nous avons rencontré un exemple d'un tel signal avec les promoteurs de *subtilis*, il en existe de nombreux autres. Pour l'instant, nous n'avons pas traité cet aspect sauf dans des cas très particuliers (comme pour les sites -10 et -35 de ces promoteurs lorsque ceux-ci sont à une distance constante l'un de l'autre — voir la section 6.2.2). Il y a au moins trois façons différentes de le faire. La première s'applique aux répétitions et aux palindromes et consiste à travailler avec nos algorithmes tels quels (version 'normale' ou ADN-X — voir section 6.2.1.2) en y ajoutant une contrainte sur la position relative des occurrences dans la chaîne. Quant aux signaux, nous pouvons, soit les identifier séparément pour les mettre ensemble ensuite, soit essayer de les caractériser de manière simultanée (avec un seul modèle). Cette seconde façon est bien sûr la plus délicate. Une approche possible du problème consisterait à introduire des jokers comme nous l'avons fait, en autorisant cependant cette fois une mise en correspondance de suites de jokers de longueurs variables ('variable length don't care symbols') à l'intérieur d'un bloc de similarité. Enfin, une approche encore plus générale exigerait l'introduction de 'méta-définitions' de ressemblance : des définitions portant cette fois sur les objets des définitions que nous avons établies dans le cadre de ce travail, c'est-à-dire, sur les modèles eux-mêmes. Nous pourrions ainsi caractériser un complexe de signaux plutôt qu'un seul signal. Ces 'méta-définitions' fourniraient alors l'équivalent des descriptions utilisées par Myers dans [Mehldau and Myers, 1993] et principalement dans [Knight and Myers, 1995] et qu'il appelle des 'super-patterns'. Dans le cas de Myers, de telles descriptions sont connues à l'avance et il s'agit d'en rechercher les in-

stances dans des séquences biologiques. Dans notre cas, il faudrait extraire les descriptions des séquences.

Obtenir des descriptions même dans les cas simples (i.e. lorsqu'il n'y a qu'un seul motif) n'est cependant pas une chose facile à faire. Ces descriptions sont pourtant importantes. En effet, la plupart du temps l'identification de signaux ou de motifs structuraux dans les macromolécules biologiques à l'aide de modèles ne représente qu'une première étape dans une étude complète de ces objets. Cette étape préliminaire doit ensuite être suivie par une utilisation des modèles trouvés afin de localiser des signaux ou motifs du même genre ailleurs. Nous avons indiqué dans le chapitre 3 (section 3.4.4) que la représentation des motifs lexicaux associés à des signaux est traditionnellement donnée, soit au moyen d'expressions régulières, soit en utilisant des matrices ou des profils. Nos modèles sont un mélange des deux, ils sont une matrice lorsqu'on ne considère que les ensembles d'occurrences, et sont des expressions régulières limitées lorsqu'on les considère comme des produits cartésiens des ensembles d'une couverture. Aucune des deux formes, matrice ou expression régulière limitée, n'est parfaite. Les matrices perdent en effet toute information concernant la dépendance entre monomères situés à des positions différentes du motif, tandis que les expressions régulières ne sont souvent pas suffisamment flexibles. En particulier, elles ne peuvent permettre de repérer que ce qui a été effectivement 'vu' dans les exemples ayant servi à sa construction. Une matrice ou un profil peuvent être ajustés (à travers des régularisateurs par exemple) pour tenir compte de ce qui n'est pas réellement observé mais que l'on suppose seulement être possible. Par ailleurs, les algorithmes, y compris les notres, biaisent en faveur des motifs les plus forts. Or nous avons vu avec les CRP (section 6.2.3) qu'un signal peut être 'délibérément' faible. Il est donc essentiel de pouvoir en tenir compte, ce que nos modèles ne sont pas capables de faire pour l'instant. Ils représentent le centre d'une boule et peuvent indiquer seulement (lorsqu'ils sont 'corrects') que les signaux recherchés doivent se trouver quelque part à l'intérieur de la boule. Dans la plupart des cas cependant, la boule est creuse : le signal ne peut être 'trop bon', c'est-à-dire, en termes pratiques, trop ressemblant au modèle.

Reste enfin le problème du tri des modèles obtenus, quels qu'ils soient, afin de séparer 'l'ivraie du bon grain'. La pertinence d'un modèle est bien sûr de nature purement biologique et, dans ce sens, nous ne pouvons rien faire au niveau des algorithmes. Cependant, il est possible d'en obtenir une première évaluation statistique. Nous en avons proposée une dans le chapitre précédent. Nous avons alors également expliqué notre sentiment concernant le moment où cette évaluation devait être réalisée, à savoir *a posteriori* et non en même temps que la recherche des modèles comme cela est le cas pour d'autres approches. Ce sentiment n'a pas changé, cependant nous pensons que l'algorithme que nous avons élaboré (CoSamp — section 6.3.2) n'est pas encore suffisamment mûr conceptuellement. Outre le fait que l'implémentation peut être rendue plus efficace, nous ne sommes pas complètement satisfaits de la stratégie adoptée. Cette insatisfaction est indépendante du problème du choix de certaines données ou paramètres (comme celui des ensembles exemples et tests), dans lesquels sans doute seule l'intuition biologique peut intervenir. Il serait intéressant également de voir dans quelle mesure l'étape d'évaluation ne pourrait être retardée. Notre intuition est qu'il est possible de pousser l'exploration combinatoire plus loin et d'obtenir des résultats plus fins avant d'avoir à les évaluer. Il y a ainsi des facteurs que nous n'essayons pas d'optimiser (par exemple, le quorum par rapport à la longueur des modèles) et qui nous permettraient peut-être d'éliminer des 'mauvais' modèles sur la base de critères non nécessairement statistiques. Observons en outre que la méthode telle qu'actuellement conçue ne permet en fait d'évaluer que des modèles très simples, en particulier sans trous, et qu'elle ne résoud pas le problème de la 'condensation' des modèles à quelque niveau que ce soit.

7.2 Les critiques et perspectives portant sur l'aspect biologique

Les critiques et perspectives de notre travail portant sur les questions de biologie se réfèrent aux définitions de ressemblance que nous avons établies et qui, malgré leur diversité, restent parfois insuffisantes. Nous allons en donner ici plusieurs exemples.

Le premier concerne la notion de comparaison multiple. Considérons le cas de la valeur à attribuer à un bloc $\mathcal{B}$. Supposons que nous décidons d'affecter à $\mathcal{B}$ un score SP, c'est-à-dire, sa 'valeur' est égale à la somme des scores SP de chacune des colonnes du bloc, où le score SP d'une colonne est lui-même la somme des scores de toutes les paires de monomères que la colonne comporte. Supposons également que nous disposons de deux blocs, en fait, de deux modèles M et M' ayant même scores SP mais tels que M est de longueur k et est présent dans $\frac{N}{2}$ chaînes distinctes tandis que M' est de longueur $2k$ et est présent dans $\frac{N}{4}$ chaînes distinctes. Lequel des deux est le plus pertinent? Une évaluation statistique, si tant est qu'elle soit possible, ne nous permettrait de répondre à cette question que dans le cadre d'un modèle dont la validité est toujours sujette à caution. Par ailleurs, si nous changeons la définition du score multiple, par exemple si nous adoptons des scores 'Étoile Centrale' (section 3.4.1.5) à la place des scores SP, il est possible que les deux blocs ne présentent plus le même score et la réponse à la question de la pertinence peut changer. Lorsque nous comparons un ensemble de chaînes relativement homogène dans lequel ce que nous recherchons doit se trouver présent partout ou presque, nous pouvons, soit considérer que le modèle le plus long est celui qui nous intéresse le plus (c'est que ce nous avons souvent fait dans les illustrations — par exemple pour les CRP et les IRE), soit envisager d'optimiser à la fois sur la longueur des modèles et sur le quorum ainsi que nous l'avons suggéré pour CoSamp dans la section précédente. Il existe toutefois au moins une situation pour laquelle il nous est pour l'instant algorithmiquement impossible de décider lequel des modèles M et M' donnés en exemple ci-dessus est le 'bon'. Cette situation est celle d'un criblage multiple de banque.

Rappelons en effet que le problème du criblage est à la fois plus simple et plus délicat à traiter que celui d'une analyse multiple portant sur un ensemble plus ou moins homogène de chaînes. Il est plus simple car il ne s'agit plus ici de localiser précisément les régions qui se trouvent impliquées dans une fonction biologique, encore moins d'essayer de comprendre de manière plus fine le mécanisme de reconnaissance qui permet à une molécule d'établir les contacts aux travers lesquels elle exerce son activité. Le but du criblage est 'uniquement' de savoir, de façon même grossière, si une molécule nouvellement séquencée ressemble au moins partiellement à quelque chose de déjà connu. Par contre, le problème qu'un tel criblage pose est plus compliqué pour au moins deux raisons. La première est de nature algorithmique. En effet, les algorithmes introduits dans ce travail ne supportent, à l'heure actuelle, qu'un nombre relativement restreint de chaînes ((jusqu'à 1000 environ). Ce nombre est très inférieur à ce que contient une banque complète (par exemple, 52 000 chaînes pour une banque de protéines). La seconde raison, probablement plus importante, est qu'une banque est un ensemble très hétérogène de chaînes. Par hétérogène, nous entendons ici que, d'une part, la banque contient de nombreuses chaînes qui n'ont rien à voir entre elles (et surtout n'ont rien à voir avec la chaîne requête) et, d'autre part, qu'elle est souvent très redondante, c'est-à-dire, qu'elle contient de multiples exemplaires ou variantes d'une même protéine. Enfin, les banques sont généralement biaisées, dans la mesure où les familles de protéines qui y sont stockées ne constituent pas nécessairement un échantillonnage représentatif de celles qui sont exprimées chez les différents organismes. Si la chaîne requête dont la fonction est inconnue est vraiment 'nouvelle', c'est-à-dire, si elle n'a pas de 'parent' très proche dans la banque, alors l'établissement de son identité, et donc de sa fonction, en présence d'autant de bruit est difficile. L'utilisation de comparaisons

multiples, plutôt que deux à deux, permet d'amplifier les faibles ressemblances entre objets mais cet avantage s'effondre d'une certaine façon lorsque ces comparaisons portent sur un aussi grand nombre d'objets trop hétérogènes. En effet, les similarités qui dénotent de 'vraies' homologies se retrouvent alors très souvent noyées parmi les 'fausses', et il devient très difficile de les départager. Ce problème n'est bien sûr pas seulement lié au fait qu'il y a plus d'objets à comparer, ou à la nature de ces objets, mais est également associé au système de score choisi. En particulier, il ne se pose pas dans le cas des structures tridimensionnelles parce que l'extension d'une distance entre deux objets se fait de manière naturelle et intuitive au cas de plus de deux objets. Cela n'est plus vrai pour les séquences. Le prototype de cribleur multiple que nous avons développé (et qui constitue une extension naturelle des algorithmes introduits dans ce travail) 'marche' ainsi relativement bien en tant qu'algorithme, mais produit des résultats biologiques insatisfaisants car il n'est pas capable de distinguer, parmi tous les modèles qu'il trouve, lesquels apportent une information quant à l'activité de la chaîne requête, et lesquels ne sont dus qu'au hasard. Deux voies paraissent possibles pour essayer de contourner cet obstacle. La première consiste à mettre en place un système d'évaluation des modèles obtenus qui serait plus adapté au type de problème rencontré lors d'un criblage. La seconde exige une remise en cause des définitions de ressemblance pour le cas spécifique du criblage, en particulier la notion de distance ou de similarité entre plus de deux objets à la fois. Aucun des systèmes proposés à l'heure actuelle (scores SP, distance de Levenshtein ou autre) ne semble réellement adéquat.

L'inadéquation de certaines des mesures de ressemblance adoptées dans ce travail ne s'applique pas qu'aux cribleurs. Considérons par exemple la question des trous. Le problème n'est pas seulement de savoir quelle valeur numérique affecter à la présence d'un trou dans un alignement global ou à l'intérieur d'un bloc, et en particulier comment compter les trous qui se succèdent; le problème est plutôt de savoir comment 'interpréter' ces trous. Dans les études phylogénétiques, les trous sont en général interdits parce que, fondamentalement, on ne sait pas quelle signification accorder à de tels événements mutationels. Lorsque ceux-ci sont autorisés, ils sont traités de la même manière que les substitutions, et les algorithmes cherchent simplement à minimiser le nombre de ces événements. Il a été montré cependant que cette minimisation de la distance entre deux séquences en termes de mutations peut produire des alignements qui sont souvent inconsistants avec leur similarité structurale [Creighton, 1993], pour ne pas parler de leur histoire évolutive. Ce problème devient d'autant plus aigu que nous cherchons à aligner non plus deux, mais plus de deux séquences simultanément puisque dans ce cas nous ne disposons même plus d'une notion de distance aussi intuitivement claire que pour le deux à deux. Pourtant, il est important de pouvoir tenir compte des trous, y compris dans les études phylogénétiques. Nous ne pouvons cependant pas le faire de la façon habituelle. Ainsi, Barriel [Barriel, 1994] par exemple suggère de considérer de manière différente la présence d'une suite de plusieurs trous dans une des chaînes d'un ensemble que l'on aligne de manière multiple suivant que d'autres types de mutations ont été ou non observées aux positions placées en correspondance avec eux sur les autres chaînes de l'ensemble étudié. Son idée est que tout système de poids doit pouvoir tenir compte d'événements emboîtés, c'est-à-dire, de mutations subséquentes à l'insertion ou délétion de plusieurs nucléotides à un endroit donné sur une des séquences.

Ce traitement différent à accorder à un trou selon que celui-ci est isolé, fait partie d'une suite contiguë d'autres trous ou se trouve dans des zones très 'mutables' de la chaîne, nous rappelle également que nous ne travaillons en fait qu'avec une partie des pièces du puzzle biologique. En ne considérant que les informations 'élémentaires' (les mutations ponctuelles), nous négligeons le fait que les objets biologiques se transforment avec le temps par des opérations portant sur des 'morceaux' plus grands de ces objets : duplication, inversion ou translocation d'un segment entier d'ADN etc. La difficulté dans ce cas n'est pas seulement d'élaborer des algorithmes qui

puissent traiter ces transformations, mais est surtout de formaliser leur nature exacte, les règles auxquelles elles obéissent et l'importance à accorder à chacune.

Les liens évolutifs entre macromolécules en général sont un autre aspect qui n'a pu être exploité jusqu'à présent que de manière heuristique par les programmes d'alignement ou de comparaison de chaînes, cela même lorsque nous pouvons faire l'hypothèse que l'arbre phylogénétique est connu *a priori*. Lorsque l'arbre est inconnu et est construit avant l'alignement par des méthodes de regroupement ('clustering'), c'est à la fois cette première étape et la seconde, réalisant l'alignement ou la comparaison à partir de l'arbre, qui sont alors effectuées de manière heuristique. Par ailleurs, dans les deux cas, un tel arbre phylogénétique n'a été utilisé pour l'instant que dans le cadre d'un alignement global, jamais dans celui de la recherche de blocs de similarité. Or certaines des illustrations présentées dans le chapitre précédent (par exemple celle concernant les motifs 'Helix Turn Helix' de la section 6.3.5) semblent indiquer que la connaissance d'un tel arbre pourrait apporter une aide considérable à la recherche de blocs. Ces illustrations montrent également que, lorsque l'arbre phylogénétique n'est pas connu à l'avance (ce qui est le plus souvent le cas), il serait intéressant de pouvoir le construire en même temps que l'alignement et non préalablement à celui-ci. La raison fondamentale est que nous pouvons alors bénéficier des informations plus riches qu'apporte une comparaison multiple, les constructions d'arbre effectuées préalablement à la recherche de similarités ne pouvant en effet se faire que sur la base de comparaisons deux à deux.

Un autre problème important soulevé au cours de ce travail est, à nouveau, lié à la question de l'évaluation des modèles mais est considéré, cette fois, sous un autre angle. L'évaluation dont nous avons parlé dans la section précédente est de nature statistique. Dans le cas de CoSamp, une telle mesure a été établie directement à partir des objets obtenus (les modèles et leurs occurrences) par rapport à ce qui serait observé sur des objets aléatoires (elle aurait pu également s'appuyer sur une autre hypothèse nulle). À aucun moment cependant, nous ne profitons du fait qu'un objet peut être 'intéressant' non à cause de ce qu'il est en termes absolus, mais parce qu'il possède des caractéristiques qui le distinguent des autres objets. Watanabe [Watanabe, 1985] propose ainsi que la reconnaissance des frontières ('boundary recognition') séparant un objet de son entourage est un facteur important dans l'identification de celui-ci : *"Texture may be determined only by a statistical relation between neighboring points"*. Watanabe considère des critères statistiques, ou probabilistes, mais il est sans doute possible d'imaginer des critères plus 'déterministes'. Ceux-ci indiqueraient ce que les modèles qui nous intéressent ne peuvent pas être et établiraient ainsi les limites de l'espace où ces modèles peuvent se situer. Ces critères pourraient alors guider la recherche dès le départ sans courir le risque de perdre de l'information. Un exemple d'un tel critère (mais que nous ne savons 'quantifier' pour l'instant) pourrait être celui d'un taux de mutation (ou d'un type de conservation) différent dans les régions impliquées dans une activité biologique [Johnson and Overington, 1993]. Un tel phénomène est sans doute ce qui est observé dans les sites de fixation de la CRP (section 6.2.3) dont nous avons vu que la relative conservation possède un rôle essentiel dans le mécanisme de régulation de la transcription. Si nous sommes capables de limiter d'une certaine façon l'espace à explorer autrement que par la contrainte de quorum (voir l'algorithme Combi — section 5.4.2.2), nous ne savons pas vraiment comment reconnaître un objet par ce qui l'oppose au reste des objets de l'univers ou l'en différencie. Nos modèles sont ainsi toujours construits de manière 'positive' : leurs occurrences sont tous les mots qui vérifient une certaine propriété par rapport aux modèles et par rapport à eux seulement.

L'idée développée dans le paragraphe précédent est bien sûr liée à celle du contexte dans lequel se situe un objet. Il existe une autre notion de contexte très différente qui concerne cette fois la manière avec laquelle l'environnement peut modifier l'identité même des objets ou

leurs caractéristiques essentielles. Il a été montré, par exemple, que la propension que possède un acide aminé à être substitué par un autre acide aminé quelconque varie selon l'élément de structure secondaire où se situe le résidu. Gribskov [Gribskov, 1992] va même plus loin que cela puisqu'il suggère que les scores de substitution d'un monomère par un autre devraient tenir compte des contraintes intervenant au niveau de la structure tertiaire de la macromolécule, et donc de nature non plus locale mais globale. Nous sommes impuissants à capter ce genre d'information d'origine géométrique avec nos modèles. Même les matrices de substitutions établies sur la base d'alignements structuraux (section 3.2.2.2) n'en tiennent pas totalement compte. Observons toutefois que notre incapacité n'est pas liée à la contrainte d'ordre que nous nous sommes imposée. L'espace occupé par un monomère à l'intérieur d'une macromolécule ne dépend après tout que de ce monomère et de la molécule. Il ne dépend donc pas d'un système de référence relatif mais est une quantité que nous pourrions mesurer dans l'absolu si nous connaissions la structure cristallographique de la molécule. De la même façon, nous pourrions associer de manière absolue à chaque monomère la liste de ses 'voisins immédiats' (les monomères avec lesquels il est en contact). Considérer le contexte n'est ainsi pas la même chose que se libérer de l'ordre de succession des monomères le long de la chaîne polymérique. Le fait que nous ne le fassions pas à l'heure actuelle est une limitation des définitions que nous employons pour les modèles et pour la similarité entre modèles et mots, ce n'est pas une limitation de notre approche elle-même.

Chapitre 8

Conclusion

And ever, as the story drained
The wells of fancy dry
And faintly strove that weary one
To put the subject by,
"The rest next time -" "It is *next time!"*
The happy voices cry.

Alice in Wonderland. Lewis Carroll.

Comme cela est probablement le cas de beaucoup de travaux, celui-ci s'achève sur une note à la fois pessimiste et optimiste. Le pessimisme se trouve justifié par les diverses limitations qui ont été détaillées dans le chapitre précédent.

Ainsi que nous l'avons alors évoqué, la plupart de ces limitations ne sont pas dues à notre choix initial de ne regarder qu'à l'intérieur d'une fenêtre nous obligeant à ne considérer que l'aspect linéaire des macromolécules biologiques. Ces limitations découlent plutôt du fait que nous n'avons exploité qu'une toute petite partie de l'espace sur lequel s'ouvre cette fenêtre.

Plus que la contrainte d'ordre elle-même, c'est une vision trop étroite de la notion de ressemblance qui nous paraît diminuer la portée de ce que nos algorithmes permettent d'identifier. D'une façon générale, la plupart des méthodes d'analyse des macromolécules (ADN/ARN ou protéines) par des moyens informatiques ne considèrent ces objets que sous un seul point de vue à la fois. Il peut s'agir, soit du point de vue de la structure primaire (la séquence de nucléotides ou d'acides aminés), soit de celui de la structure secondaire ou tertiaire (les chaînes dans l'espace avec ou sans contrainte d'ordre). De plus, on ignore souvent, ou on simplifie dans le meilleur des cas, le fait que les macromolécules ont une histoire évolutive et ne peuvent être traitées isolément et indépendamment les unes des autres.

L'argument que nous voulons avancer ici est que nous devons dorénavant essayer de maintenir plusieurs points de vue simultanément. Nous pouvons imaginer au moins deux voies possibles vers ce but. L'une part d'une analyse des macromolécules en tant qu'objets tridimensionnels et introduit le point de vue de la séquence, de l'évolution ou de la fonction comme autant de contraintes qui vont lui permettre de limiter l'espace de recherche. L'autre voie part de la représentation linéaire des macromolécules (les séquences mais également les structures avec une contrainte d'ordre) et introduit d'autres points de vue (contextuels, structuraux ou évolutifs) afin d'élargir la notion de ressemblance. La première voie présente l'avantage de partir de la représentation la plus complète qu'il est possible d'avoir des macromolécules considérées en dehors de leur milieu naturel (la cellule), la seconde bénéficie du pouvoir et de la flexibilité

que lui confère la simplicité de sa représentation linéaire.

C'est cette seconde approche que nous souhaitons poursuivre pour plusieurs raisons.

La première est nous pensons que c'est l'approche la plus adaptée à la compréhension de la relation entre séquence et structure. Cette relation a été beaucoup étudiée, soit dans le cadre de la prédiction de l'une à partir de l'autre, soit dans celui de la modélisation d'une structure en particulier. Les méthodes actuelles de prédiction aussi bien que de modélisation ont un domaine d'application relativement limité. Nous souhaitons comprendre pourquoi et explorer plus précisément la nature de cette relation. La seconde raison pour laquelle nous désirons poursuivre la voie élargissant les définitions de ressemblance à partir d'une représentation linéaire provient de notre sentiment que cette représentation est en fait beaucoup plus riche qu'on ne l'imagine. Les résultats initiaux que nous avons présentés dans le chapitre consacré aux illustrations nous semblent suffisamment encourageants pour nous renforcer dans cette impression.

Cette voie doit bien sûr nous conduire à des approches dont l'aspect lexical ne résidera plus que dans l'existence d'un ordre linéaire sur les éléments composants la macromolécule. Il est clair qu'il faudra étendre la définition d'un mot bien au-delà de ce qu'il représente actuellement à notre esprit. Nous avons déjà commencé à travailler avec des modèles qui sont des objets plus complexes, établis à partir d'une couverture qui condense plusieurs relations de similarité en même temps. Nous avons également mentionné une première façon de modifier de manière plus radicale le sens du terme 'mot' lorsque nous avons introduit les 'faux mots' structuraux composés de symboles qui sont non contigus le long de la chaîne polypeptidique. Il sera sans doute nécessaire d'aller vers des représentations beaucoup plus élaborées encore.

L'aspect qui nous semble cependant le plus passionnant dans ce travail est que non seulement il nous conduit à réfléchir sur des procédés algorithmiques ou des mécanismes biologiques, mais qu'il nous oblige également à nous poser des questions plus fondamentales sur notre représentation des objets. Finalement, notre optimisme se fonde sur le fait qu'arrivés au bout d'une première exploration de la notion de ressemblance, dans ce cas des macromolécules biologiques, nous n'avons pas le sentiment d'être parvenus à la fin d'un parcours.

Bibliographie

[Aho *et al.*, 1974] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[Aho *et al.*, 1983] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[Altschul and Erickson, 1986] S. F. Altschul and B. W. Erickson. Optimal sequence alignment using affine gap costs. *Bull. Math. Biol.*, 48:603–616, 1986.

[Altschul and Lipman, 1989] S. F. Altschul and D. J. Lipman. Tree, stars, and multiple biological sequence alignment. *SIAM J. Appl. Math.*, 49:197–209, 1989.

[Altschul and Lipman, 1990] S. F. Altschul and D. J. Lipman. Protein database searches for multiple alignments. *Proc. Natl. Acad. Sci. USA*, 87:5509–5513, 1990.

[Altschul *et al.*, 1990] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[Altschul, 1989] S. F. Altschul. Gap costs for multiple sequence alignment. *J. Theor. Biol.*, 138:297–309, 1989.

[Altschul, 1991] S. F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, 219:555–565, 1991.

[Altschul, 1993] S. F. Altschul. A protein alignment scoring system sensitive at all evolutionary distances. *J. Mol. Evol.*, 36:290–300, 1993.

[Apostolico and Preparata, 1983] A. Apostolico and F. P. Preparata. Optimal off-line detection of repetitions in a string. *Theoret. Comput. Sci.*, 22:297–315, 1983.

[Argos, 1987] P. Argos. Analysis of sequence-similar pentapeptides in unrelated protein tertiary structures: strategies for protein folding and a guide for site-directed mutagenesis. *J. Mol. Biol.*, 197:331–348, 1987.

[Bacon and Anderson, 1986] D. J. Bacon and W. F. Anderson. Multiple sequence alignment. *J. Mol. Biol.*, 191:153–161, 1986.

[Bacon and Anderson, 1990] D. J. Bacon and W. F. Anderson. Multiple sequence comparison. *Meth. Enzymol.*, 183:438–447, 1990.

[Baeza-Yates and Gonnet, 1992] R. Baeza-Yates and G. H. Gonnet. A new approach to text searching. *Commun. ACM*, 35:74–82, 1992.

[Bafna *et al.*, 1994] V. Bafna, E. L. Lawler, and P. A. Pevzner. Approximation algorithms for multiple sequence alignement. In M. Crochemore and D. Gusfield, editors, *Combinatorial Pattern Matching*, volume 807 of *Lecture Notes in Computer Science*, pages 43–53. Springer-Verlag, 1994.

[Bains, 1986] W. Bains. MULTAN: a program to align multiple DNA sequences. *Nucleic Acids Res.*, 14:159–177, 1986.

[Bains, 1989] W. Bains. MULTAN (2), a multiple string alignment program for nucleic acids and proteins. *Comput. Applic. Biosc.*, 5:51–52, 1989.

[Bairoch and Boeckmann, 1991] A. Bairoch and C. Boeckmann. The SWISS-PROT protein sequence data bank. *Nucleic Acids. Res.*, 19:2247–2249, 1991.

[Bairoch, 1992] A. Bairoch. PROSITE: A dictionary of protein sites and patterns. *Nucleic Acids Res.*, 20:2013–2018, 1992.

[Bajaj and Blundell, 1984] M. Bajaj and T. Blundell. Evolution and the tertiary structure of proteins. *Annu. Rev. Biophys. Bioeng.*, 13:453–492, 1984.

[Barriel, 1994] V. Barriel. Phylogénies moléculaires et insertions-délétions de nucléotides. *C. R. Acad. Sci. Paris, Sciences de la vie*, 317:693–701, 1994.

[Barton and Sternberg, 1987a] G. J. Barton and M. J. E. Sternberg. Evaluation and improvements in the automatic alignment of protein sequences. *Protein Eng.*, 1:89–94, 1987.

[Barton and Sternberg, 1987b] G. J. Barton and M. J. E. Sternberg. A strategy for the rapid multiple alignment of protein sequences. Confidence levels from tertiary structure comparisons. *J. Mol. Biol.*, 198:327–337, 1987.

[Barton and Sternberg, 1990] G. J. Barton and M. J. E. Sternberg. Flexible protein sequence patterns: a sensitive method to detect weak structural similarities. *J. Mol. Biol.*, 212:389–402, 1990.

[Barton, 1990] G. J. Barton. Protein multiple sequence alignment and flexible pattern matching. In *Meth. Enzymol.*, volume 183, pages 403–428. Academic Press, 1990.

[Barton, 1993] G. J. Barton. An efficient algorithm to locate all locally optimal alignments between two sequences allowing for gaps. *Comput. Appl. Biosci.*, 9:729–734, 1993.

[Bashford *et al.*, 1987] D. Bashford, C. Chothia, and A. M. Lesk. Determinants of a protein fold: unique features of the globin amino acid sequence. *J. Mol. Biol.*, 212:389–402, 1987.

[Benner *et al.*, 1993] S. A. Benner, M. A. Cohen, and G. H. Gonnet. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Mol. Biol.*, 229:1065–1082, 1993.

[Benner *et al.*, 1994] S. A. Benner, M. A. Cohen, and G. H. Gonnet. Amino acid substitution during functionally constrained divergent evolution of protein sequences. *Protein Eng.*, 7:1323–1332, 1994.

[Berg and von Hippel, 1988] O. G. Berg and P. H. von Hippel. 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, 1988.

[Bernstein *et al.*, 1977] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[Billoud *et al.*, 1996] B. Billoud, M. Kontic, and A. Viari. Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence databases. *Nucleic Acids Res.*, 24:1395–1403, 1996.

[Blaisdell *et al.*, 1993] B.E. Blaisdell, K.E. Rudd, A. Matin, and S. Karlin. Significant dispersed recurrent DNA sequences in the E*scherichia coli* genome. Several new groups. *J. Mol. Biol.*, 229:833–848, 1993.

[Blanken *et al.*, 1982] R. L. Blanken, L. C. Klotz, and A. G. Hinnebusch. Computer comparison of new and existing criteria for constructing evolutionary trees from sequence data. *J. Mol. Evol.*, 19:9–19, 1982.

[Blum, 1990] N. Blum. On locally optimal alignment in genetic sequences. manuscript, 1990.

[Blumer *et al.*, 1983] A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, and R. McConnell. Linear size finite automata for the set of all subwords of a text. *Bull. Europ. Assoc. Theoret. Comput. Sci.*, 21:12–20, 1983.

[Blumer *et al.*, 1985] A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, M. T. Chen, and J. Seiferas. The smallest automaton recognizing the subwords of a text. *Theoret. Comput. Sci.*, 40:31–55, 1985.

[Botsford and Harman, 1992] J. L. Botsford and J. G. Harman. Cyclic AMP in prokaryotes. *Microbiological Rev.*, 56:100–122, 1992.

[Brutlag *et al.*, 1990] D. L. Brutlag, J-P. Dautricourt, S. Maulik, and J. Relph. Improved sensitivity of biological sequence database searches. *Comput. Applic. Biosc.*, 6:237–245, 1990.

[Brutlag *et al.*, 1993] D. L. Brutlag, J.-P. Dautricourt, R. Diaz, , J. Fier, B. Moxon, and R. F. J. Stamm. BLAZE^{TM}: an implementation of the Smith-Waterman sequence comparison algorithm on a massively parallel computer. *Computers Chem.*, 17:203–207, 1993.

[Cailliez and Pages, 1976] F. Cailliez and J. P. Pages. *Introduction à l'Analyse de Données.* Société de Mathématiques Appliquées et de Sciences Humaines, 1976.

[Carafa *et al.*, 1990] Y. D'Aubenton Carafa, E. Brody, and C. Thermes. Prediction of Rho-independent E*scherichia coli* transciption terminators. A statistical analysis of their RNA stem-loop structures. *J. Mol. Biol.*, 216:835–858, 1990.

[Cardon and Crochemore, 1982] A. Cardon and M. Crochemore. Partioning a graph in $O(\mid A \mid \log_2 \mid V \mid)$. *Theoret. Comput. Sci.*, 19:85–98, 1982.

[Cardon and Stormo, 1992] L. R. Cardon and G. D. Stormo. Expectation Maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *J. Comp. Biol.*, 223:139–170, 1992.

[Carrillo and Lipman, 1988] H. Carrillo and D. J. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48:1073–1082, 1988.

[Chen and Seiferas, 1985] M. T. Chen and J. Seiferas. Efficient and elegant subword tree construction. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume 12 of *F*, pages 97–107. NATO Advanced Science Institutes, 1985.

[Chothia, 1984] C. Chothia. Principles that determine the structure of proteins. *Annu. Rev. Biochem.*, 53:537–572, 1984.

[Chothia, 1992] C. Chothia. One thousand families for the molecular biologist. *Nature*, 357:543–544, 1992.

[Clarke and Cooke, 1992] G. M. Clarke and D. Cooke. *A Basic Course in Statistics*. Edward Arnold, 1992.

[Clarke, 1986] A. R. Clarke. Molecular chaperones in protein folding and translocation. *Curr. Opin. Struct. Biol.*, 14:141–158, 1986.

[Clift *et al.*, 1986] B. Clift, D. Haussler, R. McConnell, T. D. Schneider, and G. D. Stormo. Sequence landscapes. *Nucleic Acids Res.*, 14:141–158, 1986.

[Cobbs, 1994] A.L. Cobbs. Fast identification of approximately matching substrings. In M. Crochemore and D. Gusfield, editors, *Combinatorial Pattern Matching*, volume 807 of *Lecture Notes in Computer Science*, pages 64–74. Springer Verlag, 1994.

[Collins and Coulson, 1990] J. F. Collins and A. F. W. Coulson. Significance of protein sequence similarities. In *Meth. Enzymol.*, volume 183, pages 474–487. 1990.

[Collins *et al.*, 1988] J. F. Collins, A. F. W. Coulson, and A. Lyall. The significance of protein sequence similarities. *Comput. Appl. Biosci.*, 4:67–71, 1988.

[Combrugghe *et al.*, 1984] B. Combrugghe, S. Busby, and H. Buc. Cyclic AMP receptor protein: role in transcription activation. *Science*, 224:831–838, 1984.

[Cornette *et al.*, 1987] J. L. Cornette, K. B. Cease, H. Margalit, J. L. Spouge, J. A. Berzofsky, and C. DeLisi. Hydrophobicity scales and computational techniques for detecting amphipatic structures in proteins. *J. Mol. Biol.*, 195:659–685, 1987.

[Corpet, 1988] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Res.*, 16:10881–10890, 1988.

[Coulson *et al.*, 1987] A. F. W. Coulson, J. F. Collins, and A. Lyall. Protein and nucleic acid sequence database searching: a suitable case for parallel processing. *Comput. J.*, 30:420–424, 1987.

[Creighton, 1993] T. E. Creighton. *Proteins*. Freeman, 1993.

[Crochemore and Rytter, 1991] M. Crochemore and W. Rytter. Usefulness of the Karp-Miller-Rosenberg algorithm in parallel computations on strings and arrays. *Theoret. Comput. Sci.*, 88:59–82, 1991.

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

[Crochemore, 1981] M. Crochemore. An optimal algorithm for computing the repetitions in a word. *Inf. Proc. Letters*, 12:244–250, 1981.

[Crochemore, 1985] M. Crochemore. Optimal factor transducers. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume 12 of *F*, pages 31–43. NATO Advanced Science Institutes, 1985.

[Crochemore, 1986] M. Crochemore. Transducers and repetitions. *Theoret. Comput. Sci.*, 45:63–86, 1986.

[Danchin, 1990] A. Danchin. *Une Aurore de Pierres. Aux Origines de la Vie.* Éditions Seuil, 1990.

[Danchin, 1995] A. Danchin. 1995. communication personnelle.

[Dandekar and Hentze, 1995] T. Dandekar and M. W. Hentze. Finding the hairpin in the haystack: searching for RNA motifs. *Trends Genet.*, 11:45–50, 1995.

[Dandekar *et al.*, 1991] T. Dandekar, R. Stripecke, N. K. Gray, B. Goossen, A. Constable, H. E. Johansson, and M. W. Hentze. Identification of a novel iron-responsive element in murine and human erythroid δ-amino-levulinic acid synthase mRNA. *EMBO J.*, 10:1903–1909, 1991.

[Daune, 1993] M. Daune. *Biophysique Moléculaire. Structures en Mouvement.* InterÉditions, 1993.

[Dayhoff and Eck, 1972] M. O. Dayhoff and R. V. Eck. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*, pages 89–99. Natl. Biomed. Res. Found., Washington, 1972.

[Dayhoff *et al.*, 1978] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*, volume 5 suppl.3, pages 345–352. Natl. Biomed. Res. Found., 1978.

[Dayhoff *et al.*, 1983] M. O. Dayhoff, W. C. Barker, and L. T. Hunt. Establishing homologies in protein sequences. In *Meth. Enzymol.*, volume 91, pages 524–544. Academic Press, 1983.

[Doolittle, 1981] R. F. Doolittle. Similar amino acid sequences: chance or common ancestry. *Science*, 214:149–159, 1981.

[Doolittle, 1986] R. F. Doolittle. *Of URFs and ORFs: a primer on how to analyze derived amino acid sequences.* University Science Books, 1986.

[Duret *et al.*, 1994] L. Duret, D. Mouchiroud, and M. Gouy. HOVERGEN: a database of homologous vertebrate genes. *Nucleic Acids Res.*, 22:2360–2365, 1994.

[Duret, 1995] L. Duret. *Évolution des séquences non-codantes chez les vertébrés : Étude des contraintes fonctionnelles et structurales par analyse comparative de gènes homologues.* PhD thesis, 1995. Thèse de doctorat - Université Claude Bernard - Lyon I.

[Ebright, 1993] R. H. Ebright. Transcription activation at Class I Cap-dependent promoters. *Molecular Microbiology*, 8:797–802, 1993.

[Escalier *et al.*, 1996] V. Escalier, J. Pothier, H. Soldano, and A. Viari. Pairwise and multiple identification of three dimensional common substructures in proteins. 1996. submitted to J. Comp. Biol.

[Fay, 1992] P. Fay. Oxygen relations of nitrogen fixation in Cyanobacteria. *Microbiol. Rev.*, 56:340–373, 1992.

[Feng and Doolittle, 1987] D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 25:351–360, 1987.

[Feng and Doolittle, 1990] D.-F. Feng and R. F. Doolittle. Progressive alignment and phylogenetic tree construction of protein sequences. *Meth. Enzymol.*, 183:375–402, 1990.

[Feng *et al.*, 1985] D.-F. Feng, M. S. Johnson, and R. F. Doolittle. Aligning amino acids sequences: comparison of commonly used methods. *J. Mol. Evol.*, 21:112–125, 1985.

[Fickett, 1984] J. W. Fickett. Fast optimal alignment. *Nucleic Acids Res.*, 12:175–179, 1984.

[Finkelstein *et al.*, 1993] A. V. Finkelstein, A. M. Gutun, and A. Y. Badretdinov. Why are the same protein folds used to perform different functions? *FEBS*, 325:23–28, 1993.

[Fitch and Smith, 1983] W. M. Fitch and T. F. Smith. Optimal sequence alignments. *Proc. Natl. Acad. Sci. USA*, 80:1382–1386, 1983.

[Fitch, 1966] W. M. Fitch. An improved method of testing for evolutionary homology. *J. Mol. Biol.*, 16:9–16, 1966.

[Fitch, 1967] W. M. Fitch. Construction of phylogenetic trees. *Science*, 15:299–304, 1967.

[Fredman, 1984] M. L. Fredman. Algorithms for computing evolutionary similarity measures with length independent gap penalties. *Bull. Math. Biol.*, 46:553–566, 1984.

[Freifelder, 1990] D. Freifelder. *Biologie Moléculaire*. Masson, 1990.

[Galas *et al.*, 1985] D. J. Galas, M. Eggert, and M. S. Waterman. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from E*scherichia coli*. *J. Mol. Biol.*, 186:117–128, 1985.

[Galil and Giancarlo, 1989] Z. Galil and R. Giancarlo. Speeding up dynamic programing with application to molecular biology. *Theoret. Comput. Sci.*, 64:107–118, 1989.

[Garey and Johnson, 1979] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[Gascuel and Danchin, 1986] O. Gascuel and A. Danchin. Protein export in prokaryotes and eukaryotes: indications of a difference in the mechanism of exportation. *J. Mol. Evol.*, 24:130–142, 1986.

[Gaston *et al.*, 1990] K. Gaston, A. Bell, A. Kolb, H. Buc, and S. Busby. Stringent spacing requirements for transcription activation by CRP. *Cell*, 62:733–743, 1990.

[Gautheret *et al.*, 1990] D. Gautheret, F. Major, and R. J. Cedergren. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *Comput. Appl. Biosci.*, 6:325–331, 1990.

[Gerstein and Altman, 1995] M. Gerstein and R. B. Altman. Using a measure of structural variation to define a core for the globins. *Comput. Appl. Biosci.*, 11:633–644, 1995.

[Gething and Sambrook, 1992] M.-J. Gething and J. Sambrook. Protein folding in the cell. *Nature*, 355:33–45, 1992.

[Goldstein and Waterman, 1994] L. Goldstein and M. S. Waterman. Approximations to profile score distributions. *J. Comp. Biol.*, 1:93–104, 1994.

[Gombrich, 1987] E. H. Gombrich. *L'Art et l'Illusion. Psychologie de la Représentation Picturale*. Éditions Gallimard, 1987.

[Gonnet *et al.*, 1992] G. H. Gonnet, M. A. Cohen, and S. A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[Gonnet *et al.*, 1994] G. H. Gonnet, M. A. Cohen, and S. A. Benner. Analysis of amino acid substitution during divergent evolution: the 400 by 400 dipeptide substitution matrix. *Biochem. Biophys. Res. Commun.*, 199:489–496, 1994.

[Gotoh, 1982] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[Gotoh, 1986] O. Gotoh. Alignment of three biological sequences with an efficient traceback procedure. *J. Theor. Biol.*, 121:327–337, 1986.

[Gotoh, 1990] O. Gotoh. Optimal sequence alignment allowing for long gaps. *Bull. Math. Biol.*, 52:359–373, 1990.

[Grantham, 1974] R. Grantham. Amino acid difference formula to help explain protein evolution. *Science*, 185:862–864, 1974.

[Gribskov *et al.*, 1987] M. Gribskov, M. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, 84:4355–4358, 1987.

[Gribskov *et al.*, 1988] M. Gribskov, M. Homyak, J. Edenfield, and D. Eisenberg. Profile scanning for three-dimensional structural patterns in protein sequences. *Comput. Appl. Biosci.*, 4:61–66, 1988.

[Gribskov *et al.*, 1990] M. Gribskov, R. Luthy, and D. Eisenberg. Profile analysis. *Meth. Enzymol.*, 183:146–159, 1990.

[Gribskov, 1992] M. Gribskov. The language metaphor in sequence analysis. *Comput. Chem.*, 16:85–88, 1992.

[Gupta *et al.*, 1995] S. K. Gupta, J. D. Kececioglu, and A. A. Schaffer. Making the shortest-paths approach to sum-of-pairs multiple sequence alignment more space efficient in practice. In Z. Galil and E. Ukkonen, editors, *Combinatorial Pattern Matching*, volume 937 of *Lecture Notes in Computer Science*, pages 128–143. Springer Verlag, 1995.

[Gusfield *et al.*, 1992] D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. pages 432–439. Proc. of the Third Annual ACM-SIAM Symp. on Discrete Algorithms, 1992.

[Gusfield, 1993] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.*, 55:141–154, 1993.

[Gusfield, 1995] D. Gusfield. Algorithms on strings: A dual view from computer science and computational molecular biology. manuscript - presented at 6th Annual Combinatorial Pattern Matching - Espoo, Finland, 1995.

[Hasemann *et al.*, 1995] C. A. Hasemann, R. G. Kurumbail, S. S. Boddupalli, J. A. Peterson, and J. Deisenhofer. Structure and function of cytochromes P450: a comparative analysis of three crystal structures. *Curr. Opin. Struct. Biol.*, 2:41–62, 1995.

[Helmann, 1995] J. D. Helmann. Compilation and analysis of B*acillus subtilis* α-dependent promoter sequences: evidence for extended contact between RNA polymerase and upstream promoter DNA. *Nucleic Acids Res.*, 23:2351–2360, 1995.

[Henikoff and Henikoff, 1991] S. Henikoff and J. G. Henikoff. Automatic generation of protein blocks for database searching. *Nucleic Acids Res.*, 19:6565–6572, 1991.

[Henikoff and Henikoff, 1992] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices for protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[Henikoff, 1994] S. Henikoff. Comparative sequence analysis: finding genes. In D. W. Smith, editor, *Biocomputing. Informatics and Genome Projects*, pages 87–118. Academic Press, 1994.

[Heringa and Argos, 1993] J. Heringa and R. Argos. A method to recognize distant repeats in protein sequences. *Proteins: struct., funct., and genetics*, 17:391–411, 1993.

[Heringa, 1994] J. Heringa. The evolution and recognition of protein sequence repeats. *Comput. Chem.*, 18:233–244, 1994.

[Hertz *et al.*, 1990] G. Z. Hertz, G. W. Hartzell, and G. D. Stormo. Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci.*, 6:81–92, 1990.

[Higgins and Sharp, 1988] D. G. Higgins and P. M. Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–244, 1988.

[Higgins *et al.*, 1992] D. G. Higgins, A. J. Bleasby, and R. Fuchs. CLUSTAL V: improved software for multiple sequence alignment. *Comput. Appl. Biosci.*, 8:189–191, 1992.

[Hogeweg and Hesper, 1984] P. Hogeweg and B. Hesper. The alignment of sets of sequences and the construction of phylogenetic trees. An integrated method. *J. Mol. Evol.*, 20:175–186, 1984.

[Howard and Rees, 1994] J. B. Howard and D. C. Rees. Nitrogenase: A nucleotide-dependent molecular switch. *Ann. Rev. Biochem.*, 94:235–264, 1994.

[Hui, 1992] L. C. K. Hui. Color set size problem with applications to string matching. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 230–243. Springer-Verlag, 1992.

[Jacob, 1970] F. Jacob. *La Logique du Vivant. Une Histoire de l'Hérédité.* Éditions Gallimard, 1970.

[Jacob, 1981] F. Jacob. *Le Jeu des Possibles.* Éditions Fayard, 1981.

[Jacob, 1987] F. Jacob. *La Statue Intérieure*. Éditions Odile Jacob, 1987.

[Jean *et al.*, 1996] P. Jean, J. Pothier, P. Dansette, D. Mansuy, and A. Viari. Automated multiple analysis of protein structures: application to homology modeling of cytochromes p450. 1996. submitted to *Proteins: struct., funct., and genetics*.

[Johnson and Doolittle, 1986] M. S. Johnson and R. F. Doolittle. A method for the simultaneous alignment of three or more amino acid sequences. *J. Mol. Evol.*, 23:267–278, 1986.

[Johnson and Overington, 1993] M. S. Johnson and J. P. Overington. A structural basis for sequence comparisons. An evaluation of scoring methodologies. *J. Mol. Biol.*, 233:716–738, 1993.

[Jones *et al.*, 1992] D. T. Jones, W. R. Taylor, and J. M. Thornton. The rapid generation of mutation data matrices from protein sequences. *Comput. Appl. Biosci.*, 8:275–282, 1992.

[Kabsch and Sander, 1983] W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[Karlin and Altschul, 1990] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA*, 87:2264–2268, 1990.

[Karlin and Altschul, 1993] S. Karlin and S. F. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc. Natl. Acad. Sci. USA*, 90:5873–5877, 1993.

[Karlin and Ghandour, 1985a] S. Karlin and G. Ghandour. Multiple alphabet amino-acid sequence comparisons of the immunoglobulin κ-chain constant domain. *Proc. Natl. Acad. Sci. USA*, 82:8597–8601, 1985.

[Karlin and Ghandour, 1985b] S. Karlin and G. Ghandour. The use of multiple alphabets in κ-gene immunoglobulin DNA sequence comparisons. *The EMBO J.*, 4:1217–1223, 1985.

[Karlin *et al.*, 1988a] S. Karlin, M. Morris, G. Ghandour, and M.-Y. Leung. Algorithms for identifying local molecular sequence features. *Comput. Appl. Biosci.*, 4:41–51, 1988.

[Karlin *et al.*, 1988b] S. Karlin, M. Morris, G. Ghandour, and M.-Y. Leung. Efficient algorithms for molecular sequence analysis. *Proc. Natl. Acad. Sci. USA*, 85:841–845, 1988.

[Karp *et al.*, 1972] R. M. Karp, R. E. Miller, and A. L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. pages 125–136. Proc. 4th Annu. ACM Symp. Theory of Computing, 1972.

[Karplus and McCammon, 1983] M. Karplus and J. A. McCammon. Dynamics of proteins: elements and function. *Annu. Rev. Biochem.*, 53:263–300, 1983.

[Karplus *et al.*, 1987] M. Karplus, A. T. Brunger, R. Elber, and J. Kuriyan. Molecular dynamics: applications to proteins. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume LII, pages 381–390. NATO Advanced Science Institutes, 1987.

[Karplus, 1995] K. Karplus. Evaluating regularizers for estimating distributions of amino acids. pages 188–196, Cambridge, England, 1995. Third International Symposium on Intelligent Systems for Molecular Biology.

[Kaufmann, 1993] S. A. Kaufmann. *The Origins of Order: Self Organization and Selection in Evolution*. Oxford University Press, 1993.

[Kececioglu, 1993] J. Kececioglu. The maximum weight trace problem in multiple sequence alignement. In Z. Galil A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 106–119. Springer-Verlag, 1993.

[Kimura, 1983] M. Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1983.

[Klotz and Blanken, 1981] L. C. Klotz and R. L. Blanken. A practical method for calculating evolutionary trees from sequence data. *J. Theor. Biol.*, 91:261–272, 1981.

[Klotz *et al.*, 1979] L. C. Klotz, N. Komar, R. L. Blanken, and R. M. Mitchell. Calculation of evolutionary trees from sequence data. *Proc. Natl. Acad. Sci. USA*, 76:4516–4520, 1979.

[Knight and Myers, 1995] J. R. Knight and E. W. Myers. Super-pattern matching. *Algorithmica*, 13:211–243, 1995.

[Kosaraju, 1994] S. R. Kosaraju. Computation of squares in a string. In M. Crochemore and D. Gusfield, editors, *Combinatorial Pattern Matching*, volume 807 of *Lecture Notes in Computer Science*, pages 146–150. Springer-Verlag, 1994.

[Koshland *et al.*, 1966] D. E. Koshland, Nemethy, and Filmer. Comparison of experimental binding data and theoretical models in proteins containing subunits. *Biochemistry*, 5:365–385, 1966.

[Krogh *et al.*, 1994] A. Krogh, M. Brown, I. S. Mian, K. Sjoelander, and D. Haussler. Hidden Markov model in computational biology. Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[Kruh, 1987] J. Kruh. *Biochimie. 1. Biologie Cellulaire et Moléculaire*. Hermann, 1987.

[Kruskal, 1956] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[Landraud *et al.*, 1989] A. M. Landraud, J. F. Avril, and P. Chretienne. An algorithm for finding a common structure shared by a family of strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):890–895, 1989.

[Lawrence and Reilly, 1990] C. E. Lawrence and A. A. Reilly. 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, 1990.

[Lawrence *et al.*, 1993] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wooton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

[Lefevre and Ikeda, 1993a] C. Lefevre and J.-E. Ikeda. Pattern recognition in DNA sequences and its application to consensus foot-printing. *Comput. Appl. Biosci.*, 9:349–354, 1993.

[Lefevre and Ikeda, 1993b] C. Lefevre and J.-E. Ikeda. The position end-set tree: A small automaton for word recognition in biological sequences. *Comput. Appl. Biosci.*, 9:343–348, 1993.

[Lefevre and Ikeda, 1994] C. Lefevre and J.-E. Ikeda. A fast word search algorithm for the representation of sequence similarity in genomic DNA. *Nucleic Acids Res.*, 22:404–411, 1994.

[Lesk and Chothia, 1980] A. M. Lesk and C. Chothia. How different amino acid sequences determine similar protein structures: The structure and evolutionary dynamics of the globins. *J. Mol. Biol.*, 136:225–270, 1980.

[Leung *et al.*, 1991] M.-Y. Leung, B. E. Blaisdell, C. Burge, and S. Karlin. An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *J. Mol. Biol.*, 221:1367–1378, 1991.

[Levenshtein, 1966] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cyber. Contr. Theory*, 10:707–710, 1966.

[Levin *et al.*, 1986] J. M. Levin, B. Robson, and J. Garnier. An algorithm for secondary structure determination in proteins based on sequence similarity. *FEBS Lett.*, 205:303–308, 1986.

[Lewin, 1994] B. Lewin. *Genes V.* Oxford University Press, 1994.

[Li and Graur, 1991] W-H. Li and D. Graur. *Fundamentals of Molecular Evolution.* Sinauer Associates, 1991.

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

[Lipman *et al.*, 1989] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA*, 86:4412–4415, 1989.

[Lorenz, 1975] K. Lorenz. *L'Envers du Miroir. Une Histoire Naturelle de la Connaissance.* Flammarion, 1975.

[Luthy *et al.*, 1991] R. Luthy, A. D. McLachlan, and D. Eisenberg. Secondary structure-based profiles: use of structure-conserving scoring tables in searching protein sequence databases for structural similarities. *Proteins: struct., funct., and genetics*, 10:229–239, 1991.

[Martinez, 1983] H. M. Martinez. An efficient method for finding repeats in molecular sequences. *Nucleic Acids Res.*, 11:4629–4634, 1983.

[Martinez, 1988] H. M. Martinez. A flexible multiple sequence alignment program. *Nucleic Acids Res.*, 16:1683–1691, 1988.

[Masek and Paterson, 1983] W. J. Masek and M. S. Paterson. How to compute string-edit distances quickly. In D. Sankoff and J. B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison*, pages 337–349. Addison-Wesley, 1983.

[McCreight, 1976] E. M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23:262–272, 1976.

[McLachlan, 1971] A. D. McLachlan. Test for comparing related amino acid sequences. Cytochrome *c* and cytochrome c_{551}. *J. Mol. Biol.*, 61:409–424, 1971.

[Médigue *et al.*, 1993] C. Médigue, A. Viari, A. Hénaut, and A. Danchin. Colibri: a functional data base for the E*scherichia coli* genome. *Microbiol. Rev.*, 57:623–654, 1993.

[Mehldau and Myers, 1993] G. Mehldau and E. W. Myers. A system for pattern matching applications on biosequences. *Comput. Applic. Biosc.*, 9:299–314, 1993.

[Miller and Myers, 1988] W. Miller and E. W. Myers. Sequence comparison with concave weighting functions. *Bull. Math. Biol.*, 50:97–120, 1988.

[Miyata *et al.*, 1979] T. Miyata, S. Miyazawa, and T. Yasunaga. Two types of amino acid substitutions in protein evolution. *J. Mol. Evol.*, 12:219–236, 1979.

[Miyazawa and Jernigan, 1993] S. Miyazawa and R. L. Jernigan. A new substitution matrix for protein sequence searches based on contact frequencies in protein structures. *Protein Eng.*, 6:267–278, 1993.

[Monod *et al.*, 1963] J. Monod, J. P. Changeux, and F. Jacob. Allosteric proteins and cellular control mechanisms. *J. Mol. Biol.*, 6:306–329, 1963.

[Monod *et al.*, 1965] J. Monod, J. Wyman, and J.-P. Changeux. On the nature of allosteric transitions: a plausible model. *J. Mol. Biol.*, 12:88–118, 1965.

[Monod, 1970] J. Monod. *Le Hasard et la Nécessité*. Éditions du Seuil, 1970.

[Murata *et al.*, 1985] M. Murata, J. S. Richardson, and J. L. Sussman. Simultaneous comparison of three protein sequences. *Proc. Natl. Acad. Sci. USA*, 82:3073–3077, 1985.

[Murata, 1990] M. Murata. Three-way Needleman-Wunsch algorithm. *Meth. Enzymol.*, 183:365–375, 1990.

[Myers and Miller, 1988] E. W. Myers and W. Miller. Optimal alignment in linear space. *Comput. Applic. Biosci.*, 4:11–17, 1988.

[Myers, 1986] E. W. Myers. An O(ND) difference algorithm and its variations. *Algorithmica*, 1:251–266, 1986.

[Nakatsu *et al.*, 1982] N. Nakatsu, Y. Kambayashi, and S. Yajima. A longest common subsequence algorithm suitable for similar text strings. *Acta Informatica*, 18:171–179, 1982.

[Needleman and Wunsch, 1970] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[Nei, 1987] M. Nei. *Molecular Evolutionary Genetics*. Columbia University Press, 1987.

[Nelson *et al.*, 1993] D. R. Nelson, T. Kamataki, D. J. Waxman, F. P. Guengerich, R. W. Estabrook, R. Feyereisen, F. J. Gonzalez, M. J. Coon, I. C. Gunsalus, O. Gotoh, K. Okuda, and D. W. Nebert. The P450 Superfamily: Update on new sequences, gene mapping, accession numbers, early trivial names of enzymes, and nomenclature. *DNA and Cell Biology*, 12:1–50, 1993.

[Neuwald and Green, 1994] A. F. Neuwald and P. Green. Detecting patterns in protein sequences. *J. Mol. Biol.*, 239:698–712, 1994.

[Overington *et al.*, 1990] J. Overington, M. S. Johnson, A. Sali, and T. L. Blundell. Tertiary structural constraints on protein evolutionary diversity: templates, key residues and structure prediction. *Proc. R. Soc. Lond. B*, 241:132–145, 1990.

[Overington *et al.*, 1992] J. Overington, D. Donnelly, M. S. Johnson, A. Sali, and T. L. Blundell. Environment-specific amino acid substitution tables: tertiary templates and prediction of protein folds. *Protein Science*, 1:216–226, 1992.

[Pascarella and Argos, 1992] S. Pascarella and P. Argos. Analysis of insertions/deletions in protein structures. *J. Mol. Biol.*, 225:461–471, 1992.

[Pau, 1989] R.N. Pau. Nitrogenases without molybdenum. *Trends Biochem. Sci.*, 14:183–186, 1989.

[Pearson and Lipman, 1988] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.

[Pearson, 1990] W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Meth. Enzymol.*, 183:63–98, 1990.

[Pearson, 1991] W. R. Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11:635–650, 1991.

[Pevzner, 1992a] P. A. Pevzner. Multiple alignement with guaranteed error bounds and communication costs. In Z. Galil A. Apostolico, M. Crochemore and U. Manber, editors, *Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 202–210. Springer-Verlag, 1992.

[Pevzner, 1992b] P. A. Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM J. Appl. Math.*, 52:1763–1776, 1992.

[Posfai *et al.*, 1989] J. Posfai, A.S. Bhagwat, G. Posfai, and R.J. Roberts. Prediction motifs derived from cytosine methyltransferases. *Nucleic Acids Res.*, 17:2421–2435, 1989.

[Pothier, 1993] J. Pothier. 1993. communication personnelle.

[Prim, 1957] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[Ptitsyn and Volkenstein, 1986] O. B. Ptitsyn and M. V. Volkenstein. Protein structures and neutral theory of evolution. *J. Biomol. Struct. Dyn.*, 4(1):137–156, 1986.

[Ramachandran *et al.*, 1963] G. N. Ramachandran, C. Ramakrishnan, and V. Sesisakharan. Stereochemistry of polypeptide chain configurations. *J. Mol. Biol.*, 7:95–99, 1963.

[Rao, 1987] J. K. M. Rao. New scoring matrix for amino acid residue exchanges based on residue characteristic physical parameters. *Int. J. Pept. Protein Res.*, 29:276–281, 1987.

[Risler *et al.*, 1988] J.-L. Risler, M.-O. Delorme, H. Delacroix, and A. Henaut. Amino acid substitutions in structurally related proteins. A pattern recognition approach. *J. Mol. Biol.*, 204:1019–1029, 1988.

[Robson and Postgate, 1980] R. L. Robson and J. R. Postgate. Oxygen and hydrogen in biological nitrogen fixation. *Ann. Rev. Microbiol.*, 34:183–207, 1980.

[Rooman and Wodak, 1988] M. J. Rooman and S. J. Wodak. Identification of predictive sequence motifs limited by protein structure database size. *Nature*, 335:45–49, 1988.

[Rooman *et al.*, 1990] M. J. Rooman, J. Rodriguez, and S. J. Wodak. Relations between protein sequence and structure and their significance. *J. Mol. Biol.*, 213:337–350, 1990.

[Rooman *et al.*, 1992] M. J. Rooman, A. K. J.-P., and S. J. Wodak. Extracting information on folding from the amino acid sequence: accurate predictions for protein regions with preferred conformation in the absence of tertiary interactions. *Biochemistry*, 31:10226–10238, 1992.

[Russell, 1992] P. J. Russell. *Genetics*. Harper/Collins, 1992.

[Sagot and Viari, 1996] M.-F. Sagot and A. Viari. A double combinatorial approach to discovering patterns in biological sequences. In D. Hirschberg and G. Myers, editors, *Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 186–208. Springer-Verlag, 1996.

[Sagot *et al.*, 1995a] M.-F. Sagot, V. Escalier, A. Viari, and H. Soldano. Searching for repeated words in a text allowing for mismatches and gaps. pages 87–100, Viñas del Mar, Chili, 1995. Second South American Workshop on String Processing.

[Sagot *et al.*, 1995b] M.-F. Sagot, A. Viari, J. Pothier, and H. Soldano. Finding flexible patterns in a text - an application to 3D molecular matching. *Comput. Appl. Biosci.*, 11:59–70, 1995. presented at First International IEEE Workshop on Shape and Pattern Matching in Computational Biology, Seattle, Washington, USA.

[Sagot *et al.*, 1995c] M.-F. Sagot, A. Viari, and H. Soldano. A distance-based block searching algorithm. pages 322–331, Cambridge, England, 1995. Third International Symposium on Intelligent Systems for Molecular Biology.

[Sagot *et al.*, 1995d] M.-F. Sagot, A. Viari, and H. Soldano. Multiple comparison: a peptide matching approach. In Z. Galil and E. Ukkonen, editors, *Combinatorial Pattern Matching*, volume 937 of *Lecture Notes in Computer Science*, pages 366–385. Springer-Verlag, 1995. to appear in *Theoret. Comput. Sci.*

[Sagot *et al.*, 1996] M.-F. Sagot, A. Viari, J. Pothier, V. Escalier, and H. Soldano. Multiple comparison in biology: some mathematical formalizations of the problem and combinatorial approaches to solve it. unpublished, 1996.

[Saitou and Nei, 1987] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[Sali and Blundell, 1990] A. Sali and T. L. Blundell. Definition of general topological equivalence in protein structures. A procedure involving comparison of properties and relationship through simulated annealing and dynamic programming. *J. Mol. Biol.*, 212:403–428, 1990.

[Sankoff and Cedergren, 1983] D. Sankoff and R. J. Cedergren. Simultaneous comparison of three or more sequences related by a tree. In D. Sankoff and J. B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison*, pages 253–263. Addison-Wesley, 1983.

[Sankoff and Kruskal, 1983] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison.* Addison-Wesley, 1983.

[Sankoff *et al.*, 1973] D. Sankoff, C. Morel, and R. J. Cedergren. Evolution of 5S RNA and the nonrandomness of base replacement. *Nature New Biology*, 245:232–234, 1973.

[Sankoff *et al.*, 1976] D. Sankoff, R. J. Cedergren, and G. Lapalme. Frequency of insertion-deletion, transversion, and transition in the evolution of 5S ribosomal RNA. *J. Mol. Evol.*, 7:133–149, 1976.

[Sankoff, 1975] D. Sankoff. Minimal mutation trees of sequences. *SIAM J. Appl. Math.*, 28:35–42, 1975.

[Santibanez and Rohde, 1987] M. Santibanez and K. Rohde. A multiple alignment program for protein sequences. *Comput. Appl. Biosci.*, 3:111–114, 1987.

[Schneider *et al.*, 1986] T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht. Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, 188:415–431, 1986.

[Schuler *et al.*, 1991] G. D. Schuler, S. F. Altschul, and D. J. Lipman. A workbench for multiple alignment construction and analysis. *Proteins*, 9:180–190, 1991.

[Sellers, 1974] P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793, 1974.

[Sheridan and Venkataraghavan, 1992] R. P. Sheridan and R. Venkataraghavan. A systematic search for protein signature sequences. *Proteins: struct., funct., and genetics*, 14:16–28, 1992.

[Smith and Smith, 1990] R. F. Smith and T. S. Smith. Automatic generation of primary sequence patterns from sets of related protein sequences. *Proc. Natl. Acad. Sci. USA*, 87:118–122, 1990.

[Smith and Smith, 1992] R. F. Smith and T. F. Smith. Pattern-Induced Multi-sequence Alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for comparative protein modelling. *Protein Eng.*, 5:35–41, 1992.

[Smith and Waterman, 1981] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[Smith *et al.*, 1990] H. O. Smith, T. M. Annau, and S. Chandrasegaran. Finding sequence motifs in groups of functionally releated proteins. *Proc. Natl. Acad. Sci. USA*, 87:826–830, 1990.

[Sneath and Sokal, 1973] H. A. Sneath and R. R. Sokal. *Numerical Taxonomy.* Freeman, 1973.

[Sobel and Martinez, 1986] E. Sobel and H. M. Martinez. A multiple sequence alignment program. *Nucleic Acids Res.*, 14:363–374, 1986.

[Soldano *et al.*, 1995] H. Soldano, A. Viari, and M. Champesme. Searching for flexible repeated patterns using a non transitive similarity relation. *Pattern Recognition Letters*, 16:233–246, 1995.

[Spouge, 1989] J. L. Spouge. Speeding up dynamic programming algorithms for finding optimal lattice graphs. *SIAM J. Appl. Math.*, 5:1552–1566, 1989.

[Spouge, 1991] J. L. Spouge. Fast optimal alignment. *Comput. Appl. Biosci.*, 7:1–7, 1991.

[Stormo, 1990] G. D. Stormo. Consensus patterns in DNA. *Meth. Enzymol.*, 183:211–221, 1990.

[Stryer, 1981] L. Stryer. *Biochemistry.* Freeman, 1981.

[Subbiah and Harrison, 1989] S. Subbiah and S. C. Harrison. A method for multiple sequence alignments with gaps. *J. Mol. Biol.*, 209:539–548, 1989.

[Tatusov and Koonin, 1994] R. L. Tatusov and E. V. Koonin. A simple tool to search for sequence motifs that are conserved in BLAST outputs. *Comput. Appl. Biosci.*, 10:0–0, 1994.

[Tatusov *et al.*, 1994] R. L. Tatusov, S. F. Altschul, and E. V. Koonin. Detection of conserved segments in proteins: Iterative scanning of sequence databases with alignment blocks. *Proc. Natl. Acad. Sci. USA*, 91:12091–12095, 1994.

[Taylor, 1986a] W. R. Taylor. The classification of amino acid conservation. *J. Theor. Biol.*, 119:205–218, 1986.

[Taylor, 1986b] W. R. Taylor. Identification of protein sequence homology by consensus template alignment. *J. Mol. Biol.*, 183:233–258, 1986.

[Taylor, 1986c] W. R. Taylor. Toward a practical grammar of protein structure: protein structure prediction by template fitting. In R. Fletterick and M. Zoller, editors, *Computer Graphics and Molecular Modeling*, pages 77–84. Cold Spring Harbor Laboratory, 1986.

[Taylor, 1987] W. R. Taylor. Multiple sequence alignment by a pairwise algorithm. *Comput. Appl. Biosci.*, 3:81–87, 1987.

[Taylor, 1989] W. R. Taylor. A template based method of pattern matching in protein sequences. *Prog. Biophys. Molec. Biol.*, 54:159–252, 1989.

[Taylor, 1990] W. R. Taylor. Hierarchical method to align large numbers of biological sequences. *Meth. Enzymol.*, 183:456–474, 1990.

[Thompson *et al.*, 1994a] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignement through sequence weighting position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[Thompson *et al.*, 1994b] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Improved sensitivity of profile searches through the use of sequence weights and gap excision. *Comput. Appl. Biosci.*, 10:19–29, 1994.

[Thornton *et al.*, 1991] J. M. Thornton, T. P. Flores, D. T. Jones, and M. B. Swindells. Prediction of progress at last. *Nature*, 354:105–106, 1991.

[Ukkonen and Wood, 1993] E. Ukkonen and D. Wood. Approximate string matching with suffix automata. *Algorithmica*, 10:353–364, 1993.

[Ukkonen, 1985a] E. Ukkonen. Algorithms for approximate string matching. *Inform. Contr.*, 64:100–118, 1985.

[Ukkonen, 1985b] E. Ukkonen. Finding approximate patterns in strings. *J. Algorithms*, 6:132–137, 1985.

[Ukkonen, 1992] E. Ukkonen. Constructing suffix trees on-line in linear time. pages 484–492. IFIP'92, 1992.

[Viari and Pothier, 1993] A. Viari and J. Pothier. *SmartMulti: a tool for the multiple alignment of protein sequences using flexible blocks.* Atelier de BioInformatique, 26, rue d'Ulm - 75005 Paris, 1993. in preparation.

[Viari, 1995] A. Viari. 1995. communication personnelle.

[Vingron and von Haesler, 1994] M. Vingron and A. von Haesler. Towards integration of multiple alignment and phylogenetic tree construction. presented at DIMACS Computational Molecular Biology Year - Sequence Alignment Workshop, November 1994.

[Vingron and Waterman, 1994] M. Vingron and M. S. Waterman. Sequence alignment and penalty choice. Reviews of concepts, case studies and implications. *J. Mol. Biol.*, 235:1–12, 1994.

[Vogt *et al.*, 1995] G. Vogt, T. Etzold, and P. Argos. An assessment of amino acid exchange matrices in aligning protein sequences: The twilight zone revisited. *J. Mol. Biol.*, 249:816–831, 1995.

[Wagner and Fischer, 1974] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21:168–178, 1974.

[Watanabe, 1985] S. Watanabe. *Pattern Recognition: Human and Mechanical.* Wiley, 1985.

[Waterman and Byers, 1985] M. S. Waterman and T. H. Byers. A dynamic programming algorithm to find all solutions in a neighborhood of the optimum. *Math. Biosci.*, 77:179–188, 1985.

[Waterman and Eggert, 1987] M. S. Waterman and M. Eggert. A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J. Mol. Biol.*, 197:723–728, 1987.

[Waterman and Perlwitz, 1984] M. S. Waterman and M. Perlwitz. Line geometries for sequence comparisons. *Bull. Math. Biol.*, 46:567–577, 1984.

[Waterman *et al.*, 1976] M. S. Waterman, T. F. Smith, and W. A. Beyer. Some biological sequence metrics. *Adv. Math.*, 20:367–387, 1976.

[Waterman *et al.*, 1984] M. S. Waterman, R. Arratia, and D. J. Galas. Pattern recognition in several sequences: consensus and alignment. *Bull. Math. Biol.*, 46:515–527, 1984.

[Waterman *et al.*, 1992] M. S. Waterman, Mark Eggert, and Eric Lander. Parametric sequence comparisons. *Proc. Natl. Acad. Sci. USA*, 89:6090–6093, 1992.

[Waterman, 1983] M. S. Waterman. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proc. Natl. Acad. Sci. USA*, 80:3123–3124, 1983.

[Waterman, 1984a] M. S. Waterman. Efficient sequence alignment algorithms. *J. Theor. Biol.*, 108:333–337, 1984.

[Waterman, 1984b] M. S. Waterman. General methods of sequence comparison. *Bull. Math. Biol.*, 46:473–500, 1984.

[Waterman, 1986] M. S. Waterman. Multiple sequence alignments by consensus. *Nucleic Acids Res.*, 14:9095–9102, 1986.

[Waterman, 1989] M. S. Waterman. Consensus patterns in sequences. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, pages 93–116. CRC Press, 1989.

[Waterman, 1990] M. S. Waterman. Consensus methods for DNA and protein sequence alignment. In *Meth. Enzymol.*, volume 183, pages 221–237. Academic Press, 1990.

[Waterman, 1995] M. S. Waterman. *Introduction to Computational Biology. Maps, Sequences and Genomes*. Chapman and Hall, 1995.

[Watson and Crick, 1953] J. D. Watson and F. H. C. Crick. Molecular structure of nucleic acids. A structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.

[Watson *et al.*, 1987] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, and A. M. Weiner. *Molecular Biology of the Gene*. Benjamin/Cummings, 1987.

[Weiner, 1973] P. Weiner. Linear pattern-matching algorithms. pages 1–11, Institute of Electrical Engineers, New York, USA, 1973. 14th IEEE Annual Symposium on Switching and Automata Theory.

[Wilbur and Lipman, 1983] W. J. Wilbur and D. J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. U.S.A.*, 80:726–730, 1983.

[Wilbur, 1985] W. J. Wilbur. On the PAM matrix model of protein evolution. *Mol. Biol. Evol.*, 2:434–447, 1985.

[Wong and Chandra, 1976] A. K. C. Wong and A. K. Chandra. Bounds for the string editing problem. *J. of the ACM*, 23:13–16, 1976.

[Wong *et al.*, 1993] A. K. C. Wong, S. C. Chan, and D. K. Y. Chiu. A multiple sequence comparison method. *Bull. Math. Biol.*, 55:465–486, 1993.

[Wu and Manber, 1992a] S. Wu and U. Manber. Agrep - a fast approximate pattern-matching tool. pages 153–162, San Francisco, CA, 1992. USENIX Technical Conference.

[Wu and Manber, 1992b] S. Wu and U. Manber. Fast text searching allowing errors. *Commun. ACM*, 35:83–91, 1992.

[Wu *et al.*, 1990] S. Wu, U. Manber, and E. W. Myers. An O(NP) sequence comparison algorithm. *Inf. Proc. Letters*, 35:317–323, 1990.

[Zuker, 1991] M. Zuker. Suboptimal sequence alignment in molecular biology. Alignment with error analysis. *J. Mol. Biol.*, 221:403–420, 1991.