Formalisme CSP (Constraint Satisfaction Problem) et localisation de motifs structurés dans les textes génomiques

Patricia Thebault

Formalisme CSP et localisation de motifs structurés dans les textes génomiques.

THÈSE

présentée et soutenue le 12 juillet 2004

pour l'obtention du

Doctorat de l'Université Toulouse III - Paul Sabatier

Ecole Doctorale Biologie Santé Biotechnologies

Spécialité : Bio-informatique

par

Patricia Thébault

Composition du jury

Rapporteurs :	Pr. Alain DENISE, Professeur, Université Paris Sud
	Dr. Alain VIARI, Directeur de recherche, INRIA Rhône-Alpes
	Pr. Eric WESTHOF, Professeur, IBMC-CNRS Strasbourg
Examinateur :	Pr. Gwennaele FICHANT, Professeur, Université Toulouse 3
Directeurs de thèse :	Dr. Christine GASPIN, Directeur de recherche, INRA Toulouse
	Dr. Thomas SCHIEX, Directeur de recherche, INRA Toulouse

Unité de Biométrie et Intelligence Artificielle – INRA, 31326 CASTANET-TOLOSAN cedex

Remerciements

C'est avec un immense plaisir que je reprends le clavier pour clore ce manuscrit par la partie dédiée aux remerciements. Je remercie Alain Denise, Eric Westhof et Alain Viari d'avoir accepté la lourde tâche d'en être les rapporteurs. Un grand merci à Gwennaele Fichant d'avoir accepté de faire partie du jury.

Je remercie Christine de m'avoir proposé cette thèse, de m'avoir communiqué son enthousiasme des ARN et de l'informatique et aussi pour son amitié et son soutien moral constant, sans lesquels je n'aurais pu mener de front tant de choses. Je tiens à exprimer toute ma reconnaissance à mes deux directeurs de thèse, Christine et Thomas, pour la façon complémentaire avec laquelle ils m'ont dirigé tout au long de ma thèse et leur implication dans le suivi et l'aboutissement de ce travail. Je les remercie tous les deux très chaleureusement pour tous leurs précieux conseils, toute l'aide scientifique qu'ils m'ont apporté et tout le temps qu'ils m'ont consacré.

Je remercie vivement David et Simon d'avoir contribué à mes travaux dans le cadre de collaborations d'équipes très enrichissante. Les nombreuses discussions durant ces dernières années ont beaucoup contribué au bon déroulement de cette thèse, et je les remercie chaleureusement de leur disponibilité et d'avoir relu mon manuscrit, et leurs remarques judicieuses ont contribué à l'élaboration de sa version finale.

Une place de choix, dans cette page, revient à mes deux coachs, Sylvain et Régis, qui (hélas) ne m'ont jamais quitté d'un oeil attentif. Le mot coach est finalement de rigueur, puisque le soutien "très soutenu" dont j'ai eu l'immense privilège d'être l'objet s'est tenu au sein du laboratoire sur un plan scientifique, mais aussi sur les terrains de sports en essayant d'attraper le ballon. Je remercie chaleureusement Régis pour sa lecture attentive de ce manuscrit qui ne serait pas ce qu'il est sans l'aide qu'il a bien voulu m'apporter et sa perpétuelle disponibilité. Je tiens également à remercier tous les rugby(wo)men qui m'ont permis au cours de ces dernières années, d'extérioriser sportivement la tension de mise à chaque thèse.

Un merci particulier à Marie-France Sagot et Julien Allali pour leur gentillesse et aide précieuse concernant les arbres de suffixes.

Ces trois années et des poussières n'auraient été ce qu'elles sont sans celles qui ont précédées, et pour toute la culture bioinformatique, techniques de travail apprises (et autres...), je remercie chaleureusement Jérôme et Farid.

Je remercie l'équipe informatique de BIA auprès desquels j'ai consommé du temps "machine" et "humain" et pour leurs nombreuses et précieuses aides en temps réel : Abde, Martin et Patrick. Je remercie également Mathias, qui bien que récemment arrivé, m'a apporté toute son aide.

Je tiens à remercier et à saluer toutes les personnes du laboratoire BIA qui m'ont encouragée, aidée et qui contribuent à donner cette ambiance très agréable et chaleureuse dans l'unité. Je remercie Annick et Christine C. pour leur gentillesse, soutien et amitiés. Si j'ai pu terminer

ce travail sous les meilleurs hospices c'est grâce aux "fées" de BIA qui d'un simple coup de baguette magique font s'évanouir bien des problèmes et fournissent des conditions de travail idéales : alors merci à Maïthe, Pascale, Jacky et Monique.

Enfin, un dernier et très grand merci à mes (beaux) parents, Patrice, Manu, Mireille, Anne-so, Stéph, Clo et Fred qui m'ont fortement soutenue en répondant (notamment) toujours présents à mes nombreuses sollicitations pour prendre soin de Sasha.

Mon dernier mot est pour Sylvain et Laurent (si près de la fin) à qui je souhaite beaucoup de courage.

À Thierry et Sasha

Table des matières

Introduction **1**

1 Contexte biologique **5**

1.1 Introduction à la biologie moléculaire 5

1.1.1 Le dogme central de la biologie moléculaire 6

1.1.2 L'ADN 7

1.1.3 l'ARN 13

1.1.4 Le dogme central - thème et variations 19

1.2 Les ARN : une famille en expansion 20

1.2.1 L'état actuel des connaissances chez *E. coli* 21

1.2.2 Nomenclature des gènes d'ARN 21

1.2.3 Diversité des ARN non codants 21

1.2.4 Le mode d'action des ARN non codants 22

1.3 Quelques exemples de familles d'ARN non codants 24

1.3.1 Les ARN impliqués dans la maturation d'autres ARN 24

1.3.2 Les ARN impliqués dans la régulation de fonctions 26

1.3.3 L'interférence ARN : une révolution en génomique fonctionnelle 27

1.4 L'ARNomique, propriétés utilisées pour la localisation des ARN 34

1.4.1 Approche thermodynamique 34

1.4.2 Détermination du contenu en (G+C) 35

1.4.3 Approche par modélisation comparative 36

1.4.4 Approche descriptive 38

1.5 L'hypothèse d'un monde à ARN 39

1.6 En résumé 41

2 Recherche de mots **43**

2.1 Les séquences nucléiques vues comme un texte 44

2.1.1 Terminologie . . . 44
2.1.2 Extraction d'information . . . 45
2.1.3 Représentation de l'information . . . 47
2.1.4 La recherche de mots . . . 50
2.2 Recherche avec pré-traitement du mot . . . 51
2.2.1 Algorithme naïf . . . 52
2.2.2 Algorithme de Boyer et Moore . . . 53
2.2.3 Algorithme de Baeza-Yates et Manber . . . 54
2.2.4 Recherche approchée . . . 57
2.3 Recherche avec pré-traitement du texte . . . 59
2.3.1 L'arbre digital des suffixes . . . 59
2.3.2 L'arbre des suffixes . . . 61
2.3.3 Recherche d'un mot dans un arbre des suffixes . . . 65
2.3.4 Tableau de suffixes . . . 67
2.4 Conclusion . . . 67

3 Étude de l'existant dans le cadre CSP 69
3.1 Définitions . . . 70
3.1.1 Motifs structurés . . . 70
3.1.2 Logiciels spécifiques ou généralistes . . . 71
3.2 Étude des logiciels généralistes dans le cadre CSP . . . 72
3.2.1 Présentation du formalisme CSP . . . 72
3.2.2 Cadre de comparaison des logiciels . . . 73
3.2.3 Langages utilisateur : variables et contraintes . . . 74
3.2.4 PatScan . . . 76
3.2.5 RnaMot . . . 77
3.2.6 RnaBob . . . 78
3.2.7 Palingol . . . 79
3.2.8 RnaMotif . . . 82
3.2.9 Cove . . . 83
3.2.10 Erpin . . . 83
3.3 Cadre de la comparaison des logiciels . . . 84
3.3.1 Définitions . . . 84
3.3.2 Données utilisées . . . 85
3.4 Recherche d'ARN de transfert . . . 86

3.4.1 Description de la molécule . . . 86
3.4.2 Les algorithmes sur mesure pour les ARNt . . . 88
3.4.3 Logiciels généralistes . . . 91
3.4.4 Résultats obtenus avec *Escherichia coli* . . . 95
3.4.5 Résultats obtenus avec *Saccharomyces cerevisiae* . . . 104
3.4.6 Discussion sur la recherche des ARNt . . . 108
3.5 Recherche de snoARN à boîtes C/D . . . 112
3.5.1 Description de la molécule . . . 112
3.5.2 Un algorithme sur mesure pour les snoARN à boîtes CD . . . 113
3.5.3 Les modèles et protocoles de recherche utilisés par les logiciels généralistes . . . 114
3.5.4 Paramétrage du logiciel spécifique . . . 115
3.5.5 Données de référence . . . 115
3.6 Analyse approfondie des résultats de la recherche de snoARN dans la séquence de *P. aerophilum* . . . 121
3.6.1 Objectif . . . 121
3.6.2 Extraction des régions inter-géniques . . . 121
3.6.3 Sélection des candidats . . . 121
3.6.4 Résultats . . . 121
3.7 Discussion . . . 124
3.8 Conclusion . . . 126
3.9 Bilan et objectifs . . . 127
3.9.1 Objectifs dans le cadre de la thèse . . . 127

4 Modélisation du problème de la recherche de motif 131
4.1 Approche par les grammaires . . . 132
4.1.1 Définition d'une grammaire . . . 132
4.1.2 Hiérarchie de Noam Chomsky . . . 133
4.1.3 Langages et analyse lexicale . . . 134
4.1.4 Analyse lexicale par automate . . . 134
4.1.5 Application des grammaires aux séquences nucléiques . . . 136
4.2 Approche par les CSP . . . 137
4.2.1 Définition d'un système de contraintes . . . 137
4.2.2 Techniques de résolution d'un CSP . . . 139
4.2.3 Application des CSP aux séquences nucléiques . . . 146

4.3 Conclusion 148

5 Formalisme CSP et recherche de motifs structurés 149

5.1 Représentation d'une occurrence 149

5.1.1 Définitions 150

5.2 Formalisation dans le cadre CSP 151

5.2.1 Les contraintes unaires 152

5.2.2 Les contraintes binaires 153

5.2.3 Les contraintes d'arité 4 153

5.2.4 Quelques exemples 155

5.3 La recherche de solutions 156

5.3.1 Choix de la méthode de filtrage 156

5.3.2 Mécanisme général de résolution du CSP 157

5.4 La propagation des contraintes 160

5.4.1 Contrainte `Est_Au_Moins_Distant` 161

5.4.2 Contrainte `Est_De_Nature` 161

5.4.3 Contrainte `Est_Palindrome` 163

5.4.4 Contrainte `Est_Duplex` 169

5.5 Prototype Milpat 173

5.5.1 Architecture 173

5.5.2 Résultats 174

5.5.3 Discussion 179

Conclusion - Perspectives 181

Bibliographie 185

Introduction

Grâce aux progrès technologiques de la dernière décennie, le séquençage du génome est une étape désormais maîtrisée. L'information rendue disponible suscite l'intérêt des biologistes dont l'un des objectifs est de comprendre les mécanismes d'expression des gènes. Cette compréhension passe par une étape essentielle de déchiffrage des données brutes qui consiste à localiser dans les séquences génomiques des régions fonctionnelles. Ces séquences étant obtenues sous forme de longs textes écrits dans un alphabet à quatre lettres, la compréhension de l'organisation de ce texte passe par son analyse. On peut donc considérer que le déchiffrage de ce texte passe par l'identification de "mots" dont l'organisation et la composition constituent les éléments de base pour leur identification.

Les premières avancées dans le décryptage des génomes ont été principalement axées sur la localisation des gènes codant pour des protéines. Pendant très longtemps les ARN ont été essentiellement considérés comme des molécules transférant l'information génétique portée par l'ADN du noyau jusqu'aux ribosomes, organites qui assurent la synthèse des protéines, avec en plus quelques ARN stables faisant partie de cette machinerie de synthèse des protéines (les ARN ribosomiques et les ARN de transfert). Au cours des dernières années, il s'est avéré qu'un très grand nombre de petits ARN non codants, dont les tailles varient de 20 à 200 nucléotides, sont présents dans tout type cellulaire. Ils exercent des fonctions variées à des niveaux très divers, tels que réplication, régulation de la transcription, maturation des ARN, régulation de la traduction. Chez les eucaryotes supérieurs, ce type de régulation transcriptionnelle et/ou traductionnelle par des petits ARN semble jouer un rôle majeur dans les processus de développement. Chez les bactéries, certains de ces ARN semblent contrôler les gènes de virulence. L'identification de ces petits ARN non-codants (ARNnc) en est à ses débuts, menée essentiellement par des voies expérimentales qui sont très laborieuses. C'est à ce niveau que se place le travail de cette thèse avec l'objectif d'étudier et de développer des approches de recherche d'ARN non codant.

La bioinformatique, en tirant parti des spécificités liées aux macromolécules biologiques, permet de proposer des représentations ou des méthodologies pour l'analyse de séquences. Les macromolécules biologiques sont composées d'un enchaînement de molécules de base représentées classiquement par les lettres d'un alphabet. Ainsi, l'analyse de séquences peut s'envisager comme un problème d'analyse de très longues chaînes de caractères. Dans ce cadre, le motif peut être défini comme un ensemble d'éléments soumis à des contraintes. Il peut représenter d'un point de vue biologique les caractéristiques (ou signature) d'une famille de macromolécules fonctionnelles.

Les motifs non structurés décrivent les propriétés de l'enchaînement des éléments de séquence primaire sans prendre en compte les caractéristiques des structures de niveau supérieur des macromolécules. La complexité de ces structures est très inégale entre les acides nucléiques

composés d'un alphabet à quatre lettres et les protéines composées d'un alphabet à vingt lettres. La structure primaire[1] d'un motif protéique est souvent suffisante pour caractériser un site biologiquement actif, même si celui-ci est ambigu à certaines positions. C'est d'ailleurs pour cette raison que l'utilisation à grande échelle de méthodes de recherche de similarités dans les banques donne des résultats intéressants dans le cadre de l'identification de motifs protéiques conservés.

Dans le cas des séquences nucléiques, du fait de la faible complexité des séquences, il est plus difficile de rechercher des motifs sur la seule base des éléments de séquence. Il est nécessaire d'ajouter d'autres critères tels que la recherche de zones symétriques ou palindromiques. Pour le cas particulier des ARN, la molécule est constituée d'un brin unique replié sur lui-même en créant des interactions entre éléments de structure primaire. La structure secondaire des ARN est connue pour être mieux conservée que la séquence. Elle est généralement prise en compte dans les outils de recherche de motifs existant.

Plusieurs approches, basées sur des représentations différentes d'un motif structuré, ont été proposées pour modéliser les chaînes moléculaires et les recherches dans les séquences génomiques :

- l'analyse lexicale basée sur une représentation concise d'un ensemble de mots ou séquences au moyen de règles courtes, appelées grammaires.
- le cadre formel des problèmes de satisfaction de contraintes (CSP) basé sur une représentation de type objet décrivant un ensemble d'éléments simples, connectés entre eux par un ensemble de dépendances pouvant être complexes.

Un langage est représenté par une grammaire, dont le rôle est de capturer les caractéristiques communes à toutes les séquences appartenant à ce langage. Il doit être à la fois spécifique et suffisamment expressif. Dans le cadre de la théorie des langages, la classification des grammaires, définie par Noam Chomsky en 1957, distingue les quatre classes suivantes en fonction de la forme des grammaires et de leur expressivité. A chacune de ces grammaires correspond un automate permettant l'analyse lexicale des mots appartenant au langage considéré [Cho57]. Le langage d'un niveau donné est inclus dans les langages de niveaux supérieurs. Les grammaires régulières et hors-contextes possèdent des parsers optimaux basés sur des algorithmes de complexité polynomiale. Le "parsing" des langages contextuels est NP-complet et celui des langages non restreints peut être indécidable.

En pratique, Searls [SD93] a démontré que le niveau des langages hors contextes capture les structures telles que les palindromes mais que les répétitions appartiennent au niveau suivant. Il a cherché une approche minimaliste pour trouver une grammaire pouvant prendre en compte les copies sans aller jusqu'au niveau des langages contextuels.

Les outils de recherche de motifs structurés s'organisent en essentiellement deux grandes classes : les logiciels spécifiques et les logiciels généralistes. Les logiciels dits famille-spécifiques sont dédiés à une famille particulière. Un outil de ce type est spécifique à une famille de molécules (ARNt, snoARN,...), et prend en considération les particularités de ces molécules. De plus, il pondère de façon spécifique la présence discriminante ou non de chaque élément du motif. Le taux de reconnaissance et d'ambiguïté d'un tel outil permet de mesurer la compréhension

1. La structure primaire donne l'ordre d'enchaînement des acides aminés ou nucléotides qui composent la protéine ou ARN. La séquence se replie ensuite dans l'espace en une structure secondaire stabilisée par des liaisons hydrogène et des liaisons hydrophobes.

que l'on a acquise du message contenu dans les séquences. Cette approche sur mesure s'avère très performante mais requiert un important travail d'adaptation pour passer d'un problème à l'autre.

Les logiciels généralistes requièrent une interaction importante avec l'utilisateur, ce dernier ayant à définir le motif à rechercher. Ils disposent d'un langage pour traduire l'ensemble des éléments et leurs caractéristiques de manière plus ou moins souple et efficace suivant le logiciel considéré, et d'un algorithme de recherche du motif, dont le rôle est de satisfaire les contraintes spécifiées par l'utilisateur. L'avantage d'un tel outil est qu'il peut en théorie s'utiliser pour rechercher tout type de motif, à condition que le langage soit suffisamment puissant pour décrire les caractéristiques du motif.

Le travail original de la thèse porte sur la proposition d'un modèle et les développements informatiques d'un outil généraliste permettant de rechercher dans les séquences génomiques des régions non codantes fonctionnelles pouvant mettre en jeu des interactions internes à la molécule mais aussi avec d'autres régions d'un génome cible. Cette recherche s'effectue sur la base d'une signature caractérisant les structures des régions fonctionnelles, ses interactions avec d'autres séquences et les positionnements relatifs de ces éléments.

La modélisation mathématique d'un problème combinatoire, par exemple la recherche de motifs structurés, est le préalable indispensable à sa résolution. Cette modélisation est souvent rendue délicate par la variété de caractéristiques qui doivent être prises en compte [SFV97] :

- *le coeur du problème peut généralement s'exprimer sous la forme de contraintes qui doivent être absolument satisfaites et qui résultent des propriétés incontournables des objets qui interviennent.*
- *à ces contraintes physiques s'ajoutent des volontés, plus ou moins fortes, de voir certaines propriétés ou contraintes satisfaites par la solution ;*
- *enfin, il faut prendre en compte le fait que le modèle mathématique réalisé ne modélise pas parfaitement ou complètement le problème posé par la réalité : faible précision des données ou impossibilité de prendre en compte certaines d'entre elles.*

Les techniques de satisfaction de contraintes (CSP pour Constraint Satisfaction Problem) permettent de traiter les situations où l'on souhaite résoudre un problème en décrivant les caractéristiques des solutions, plutôt qu'en écrivant une procédure de résolution.

Notre choix méthodologique s'est orienté vers le formalisme CSP et a donné le prototype MilPat (**M**otifs and **I**nter-mo**L**ecular motifs searching tool using cs**P** form**A**lism and solving **T**echniques) que nous décrivons dans ce manuscrit.

Structure du manuscrit

Cette thèse se divise en cinq chapitres.

Le chapitre 1 présente un état de l'art, en biologie, sur le contexte actuel de découverte d'un nombre de plus en plus grand de gènes et de motifs d'ARN impliqués dans la régulation d'un grand nombre de processus.

Le chapitre 2 présente un état de l'art des algorithmes de recherche de mots mettant en évidence les avancées dans ce domaine qui ont donné lieu à la production d'algorithmes efficaces sur lesquels nous nous sommes appuyés pour nos développements.

Le chapitre 3 développe de façon approfondie l'existant en termes d'approches informatiques et d'outils existants pour la localisation de familles d'ARN. Les outils étudiés ont été explorés de façon fouillée avec un souci d'illustration de leurs différences, leurs points communs, leurs forces et faiblesses. Une analyse fine des résultats obtenus dans le cadre de ce travail sur le génome de *Pyrobaculum aerophilum* s'est finalisée par la découverte d'environ 40 candidats sérieux de régions fonctionnelles apparentés aux snoRNA à boîtes C/D encore inconnus et (pour certains) originaux. Ces résultats sont en cours de validation expérimentale.

Le chapitre 4 porte sur la présentation et les avantages du cadre CSP (Constraint Satisfaction Problem) en tant que formalisme présentant une grande souplesse pour modéliser les interactions ARN-ARN intra et inter moléculaires.

Le chapitre 5 propose un modèle de spécification dans le cadre CSP qui permet de décrire des relations inter et intra-moléculaires avec une grande souplesse. Ce modèle et les algorithmes associés ont été développés et implémentés à travers un prototype nommé MilPat.

Chapitre 1

Contexte biologique

1.1 Introduction à la biologie moléculaire

Le génome contient l'ensemble des informations indispensables à la reproduction et au fonctionnement des organismes. Il est constitué d'une à plusieurs macromolécules d'acide désoxyribonucléique appelé ADN. Cet ADN contient les unités fonctionnelles appelées gènes ou encore unités de transcription. La transmission des caractères spécifiques s'effectue à partir du génome de chaque cellule. Les gènes sont tout d'abord transcrits en ARN (acide ribonucléique) pour être ensuite traduits en protéines. Chaque protéine possède une fonction particulière permettant le fonctionnement et la vie de chaque cellule de l'organisme.

Les ARN ont ainsi longtemps été considérés comme des intermédiaires instables dont le rôle était de transmettre l'information génétique depuis l'ADN jusqu'aux ribosomes [2] où la synthèse protéique a lieu. Récemment, des travaux en biologie ont permis de découvrir que les ARN jouent en fait des rôles multiples et essentiels dans toutes les cellules vivantes et pour beaucoup de virus, pour lesquels ils peuvent d'ailleurs être les porteurs de l'information génétique.

Les ARN ont la capacité de pouvoir directement intervenir dans les processus catalytiques (exemple : l'élimination des introns des ARN pré-messagers nucléaires, la maturation des ARN ribosomiques, la synthèse des protéines).

Ils jouent aussi un rôle fondamental dans des processus de régulation de la réplication de l'ADN, de la transcription de l'ADN et de la traduction, influençant ainsi la division et la différentiation cellulaire ainsi que le développement.

Cette capacité des ARN à intervenir dans différents processus, y compris dans la catalyse en tant que ribozyme, est liée aux propriétés complexes de repliement de l'ARN et à la réactivité de ses résidus au sein des structures complexes ainsi formées. De ce fait, l'ARN est maintenant considéré comme une cible potentielle très intéressante en pharmacologie.

Deux brins d'ADN ou d'ARN peuvent s'apparier formant une interaction d'autant plus stable que les séries de bases complémentaires sont longues. De cette propriété résulte l'organisation de l'ADN des chromosomes en une double hélice formée de deux brins complémentaires sur toute leur longueur. Les ARN, quant à eux, sont des copies de portions plus ou moins longues d'ADN et se trouvent souvent sous la forme de simple brin. De ce fait, grâce à la propriété d'in-

2. Ribosome : machinerie cellulaire, constituée de protéines et d'ARN (les ARNr), responsable de la traduction des ARNm.

teraction entre bases complémentaires, ils ont la particularité de se replier sur eux-mêmes ou de s'apparier à d'autres molécules (ADN, ARN ou protéines). La fonction des ARN est intimement liée à la structure tridimensionnelle que ces molécules adoptent et qui résulte des interactions internes ou avec d'autres molécules (ADN, ARN ou protéines).

1.1.1 Le dogme central de la biologie moléculaire

Le "dogme central de la biologie moléculaire" (Fig 1.1), introduit par Francis Crick (le co-découvreur de la structure de l'ADN) en 1958, établit la liaison qui existe entre le matériel génétique contenu dans la cellule et les protéines que cette cellule synthétise.

Le passage du gène à la protéine se fait en deux étapes : tout d'abord le segment de la molécule d'ADN correspondant au gène est copié sur un brin d'ARN (structure transitoire) que l'on appelle ARN messager (ou ARNm) : c'est la transcription. Puis ce brin d'ARNm est à son tour recopié mais dans un langage différent, celui des acides aminés, pour donner la séquence correspondant à la protéine synthétisée : c'est la traduction. La séquence polypeptidique ainsi constituée prend alors une forme spatiale qui lui est spécifique pour devenir la protéine.

Chez les eucaryotes, les gènes codant pour des protéines sont le plus souvent constitués de deux types de séquences nucléotidiques : les exons, conservés dans l'ARN messager fonctionnel après une étape d'épissage et les introns excisés au cours de cette même étape. Les parties codantes, contenues dans les exons, portent l'information qui sera utilisée pour fabriquer les protéines. Du fait de la disposition alternée des exons/introns, on emploie l'expression de gène mosaïque.

Les différents processus impliqués dans le transfert de l'information

Transfert ADN $\Rightarrow$ ADN : correspond à l'étape de **réplication** nécessaire à la conservation de l'information génétique lors de la division cellulaire.
Transfert ADN $\Rightarrow$ ARN : correspond à l'étape de **transcription**. Chez les eucaryotes, trois ARN polymérases différentes sont responsables de la synthèse des ARN, suivant le type de l'ARN (contrairement aux procaryotes où une seule enzyme existe).

- ARN polymérase I : ARNr
- ARN polymérase II : ARNm
- ARN polymérase III : ARNt, ARN 5S....

Transfert ARN $\Rightarrow$ protéine : correspond à l'étape de **traduction** permettant la synthèse des protéines grâce à l'information contenue dans les ARNm.

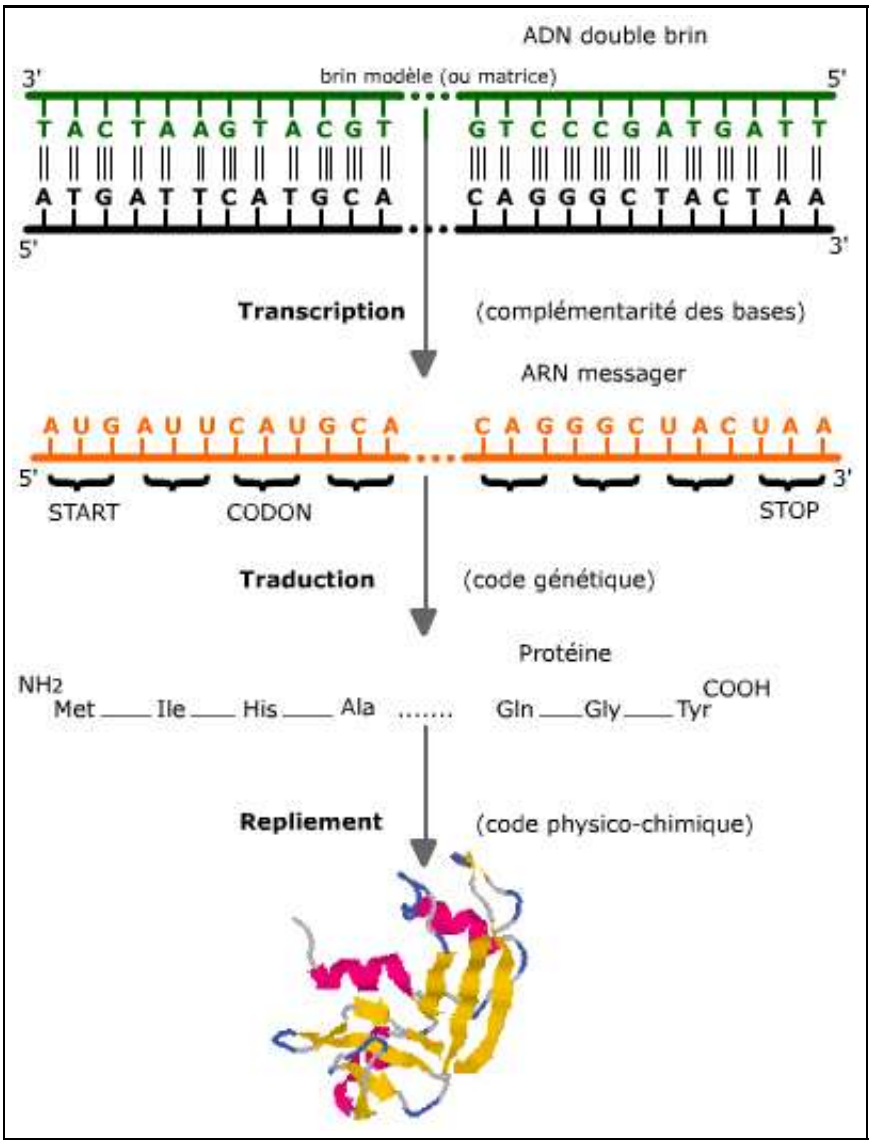


FIG. 1.1 – *Dogme central de la biologie moléculaire : Ce flux d'information décrit la transcription, par laquelle l'information contenue dans l'ADN est recopiée (transcrite) sous forme d'ARN en utilisant le principe de la complémentarité des bases et la traduction, par laquelle l'information contenue dans l'ARN est traduite en chaînes linéaires d'acides aminés selon le code génétique.*

1.1.2 L'ADN

L'information génétique est stockée dans des polymères de structure relativement simple connus sous le nom d'acides désoxyribonucléiques.

1.1.2.1 Composition chimique

L'ADN est une macro-molécule formée de l'assemblage linéaire de désoxyribonucléotides, chacun étant constitué par l'assemblage d'un sucre (désoxyribose), d'un groupement phosphate et d'une base (Fig. 1.2).

Le squelette de la molécule d'ADN est constitué de désoxyriboses unis par des ponts phosphodiester. L'hydroxyle [3] de la partie osidique d'un désoxyribonucléotide est lié par une liaison phosphodiester à l'hydroxyle de l'ose adjacent. Cette liaison asymétrique impose une orientation à la molécule et permet de définir une orientation des brins d'ADN. Les extrémités libres sont nommées 5' et 3' selon le carbone se trouvant non lié. Les séquences sont généralement représentées dans le sens 5' vers 3' (Fig. 1.3).

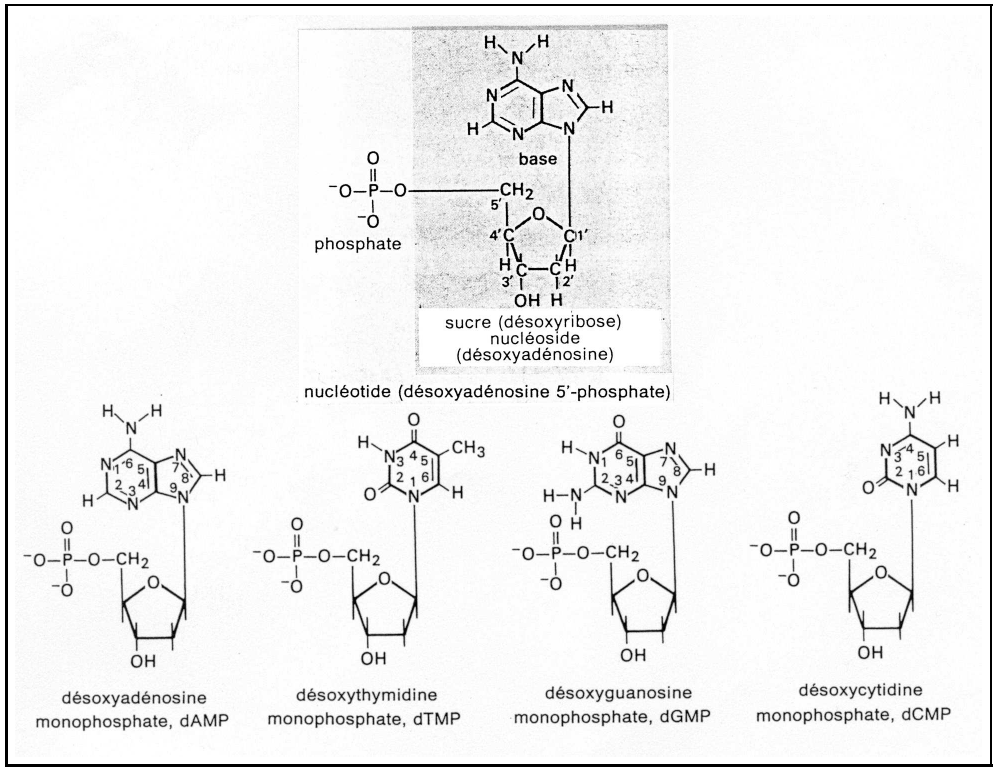


FIG. 1.2 – *Les désoxynucléotides de l'ADN (d'après [SB92]).*

La partie variable de l'ADN, porteuse de l'information génétique, est composée par les bases. Il en existe deux classes : les bases puriques (adénine et guanine) qui contiennent 2 cycles et les bases pyrimidiques (cytosine et thymine) qui ne contiennent qu'un cycle.

On représente un nucléotide par l'initiale de la base associée (A,C,G ou T). Le nucléotide est l'élément de base utilisé pour construire les acides nucléiques. Un brin d'ADN est complètement défini par l'agencement des bases (A,C,G ou T) de ses nucléotides, orienté selon un sens de lecture, et permet de considérer cette molécule comme un texte linéaire sur un alphabet à 4 lettres.

3. un hydroxyle est composé d'un atome d'oxygène et hydrogène

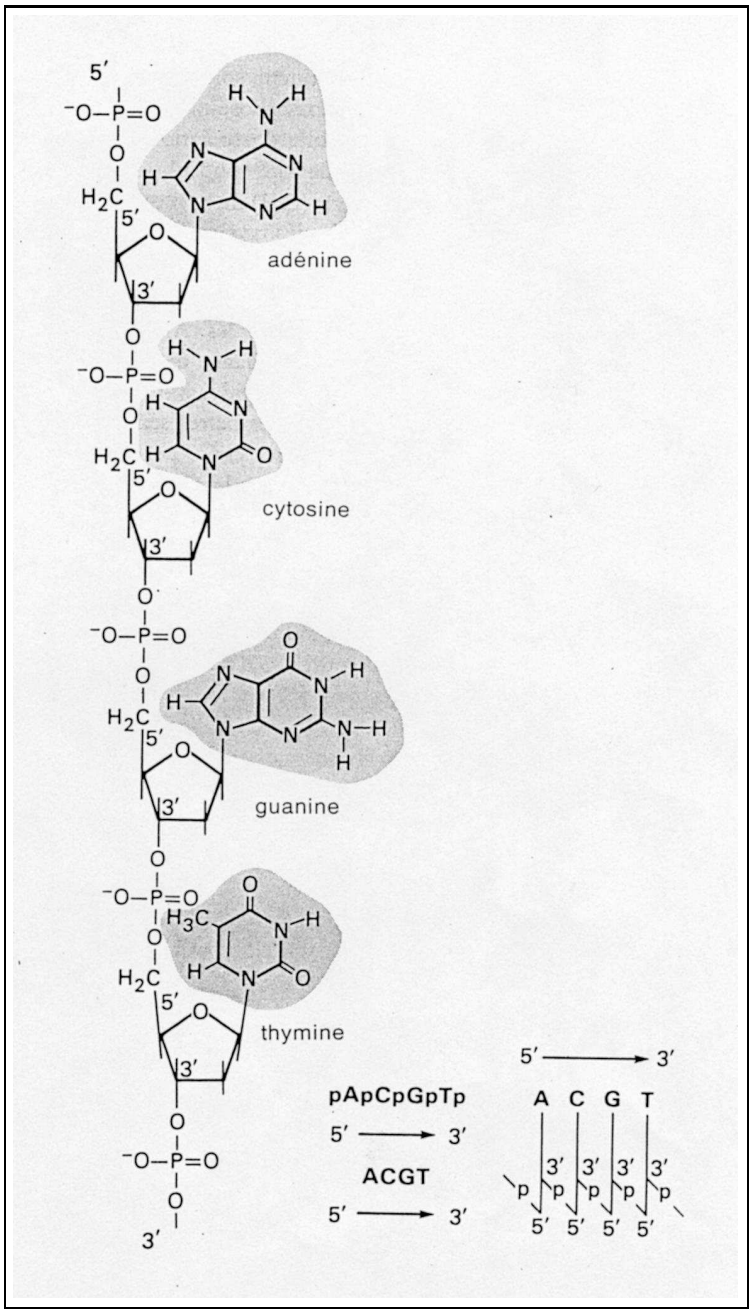


FIG. 1.3 – *Les liaisons entre désoxynucléotides adjacents dans une chaîne polynucléotidique. La partie inférieure droite de la figure montre diverses représentations schématiques d'une séquence de désoxynucléotides. Les séquences de nucléotides sont en général écrites et lues de gauche à droite, de l'extrémité 5' à 3' (d'après [SB92]).*

1.1.2.2 Structure de l'ADN

Les forces intervenant dans la structure sont des forces faibles mais qui, ensemble, assurent la stabilité des structures tridimensionnelles des acides nucléiques.

Deux types d'interactions sont possibles entre bases. Les interactions perpendiculaires aux

plans des deux bases sont appelées **forces d'empilement** et les interactions parallèles aux plans de bases, appelées **forces d'appariement**.

Forces d'appariement

La structure chimique des bases leur permet de s'apparier entre elles. Dans l'ADN, les nucléotides complémentaires sont A/T et G/C. Ces appariements nommés appariements de type Watson et Crick (Fig. 1.8) sont le résultat de liaisons hydrogènes se formant entre les bases capables de partager un proton (composant le noyau d'un hydrogène). L'union A/T contient deux liaisons hydrogènes tandis que l'appariement C/G met en jeu trois liaisons hydrogènes.

Forces d'empilement

Le deuxième type de force ayant une incidence sur la structure de la molécule est le phénomène d'empilement. Les bases ont tendance à s'empiler sous l'action de plusieurs forces dont la principale est l'hydrophobicité. La liaison hydrogène est exclue de l'empilement en lui-même. Le phénomène d'empilement est étroitement lié à la nature des bases.

1.1.2.3 La double hélice de l'ADN

Grâce à la combinaison de données issues de techniques différentes, Watson et Crick[4] ont proposé [WC53] le premier modèle en double hélice de l'ADN.

Ces travaux leur ont permis de mettre en évidence :

- l'existence de deux chaînes dans la molécule d'ADN ;
- leur association en une forme d'hélice ;
- une association anti-paralléle de ces deux chaînes.

A partir de ces résultats (deux chaînes antiparallèles en hélice associées par des bases azotées complémentaires deux à deux), Watson et Crick ont pu réaliser un modèle moléculaire tridimensionnel de la molécule d'ADN.

Les trois structures classiques des doubles brins d'ADN sont les formes (Fig. 1.4) :

A : forme rarement observée avec l'ADN, seulement en présence d'un très faible taux d'hydratation mais qui existe souvent dans l'ARN ;

B : forme la plus courante ;

Z : forme appelée zigzag, observée quelquefois. Expérimentalement on observe que de courtes molécules d'ADN constituées exclusivement de G et de C prennent spontanément prennent une forme Z [KD93].

Les formes A et B sont composées d'hélices droites emmêlées en torsade tandis que la forme Z est composée d'hélices gauches. ,

4. Prix Nobel en 1962

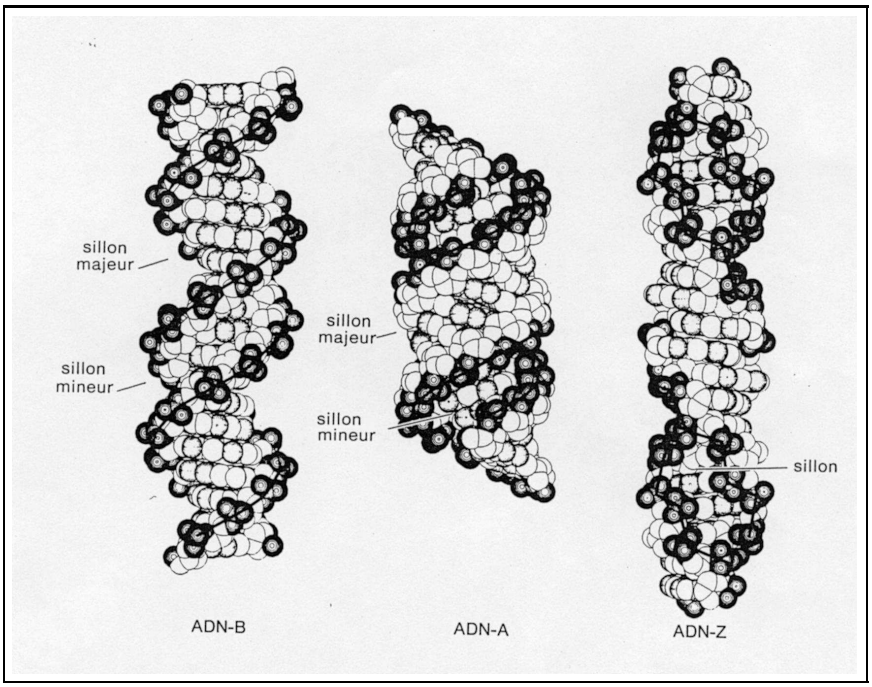


FIG. 1.4 – *Représentation compacte (Van der Waals) des ADN-B, A et Z (chacune contenant 20 paires de bases). Les atomes de phosphore et l'oxygène du phosphate sont représentés par des cercles noir et épais et les atomes d'azote sont légèrement pointillés. La ligne épaisse de phosphate à phosphate trace le squelette de la chaîne polynucléotidique. L'ADN-A est plus large et plus court, l'ADN-Z est légèrement plus long et plus étroit que l'ADN-B (d'après [SB92]).*

La structure même de la molécule permet de comprendre comment elle peut être recopiée sans erreur du fait de la complémentarité des bases et ainsi être transmise aux 2 cellules filles lors de la division cellulaire (Fig. 1.5).

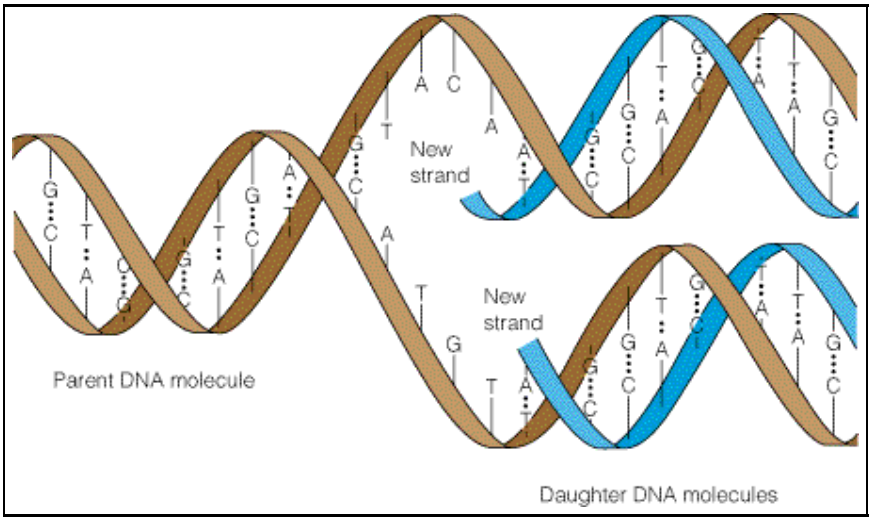


FIG. 1.5 – *Modèle pour la réplication de l'ADN : deux molécules filles (en bleu) sont répliquées à partir de la double hélice.*

1.1.2.4 Les structures atypiques de l'ADN

La double hélice n'est pas la seule structure possible de l'ADN. Il existe également des structures atypiques faisant intervenir non pas deux mais trois ou quatre brins d'ADN. Ce sont respectivement les triples hélices et les quadruplexes [ALM+00].

Les triples hélices

Une triple hélice [Gui02] stable peut être formée par une séquence de nucléotides contenant exclusivement une succession de guanines et d'adénines empilés trois par trois selon plusieurs combinaisons possibles (Fig 1.6). Les séquences alternant bases puriques et bases pyrimidiques de la double hélice défavorisent cette formation.

Les triples hélices se forment :

- soit de façon intermoléculaire via l'utilisation de nucléotides constituant ainsi la troisième chaîne,
- soit de façon intramoléculaire ; il est en effet possible, sous certaines contraintes, que la double hélice d'ADN s'ouvre localement, autorisant ainsi l'une des deux chaînes de l'hélice à se replier sur une séquence proche avec formation d'une triple hélice intramoléculaire.

Les séquences de nucléotides nécessaires à la formation d'une triple hélice sont des motifs souvent retrouvés dans les régions régulatrices des gènes et peuvent être reconnues par des facteurs de transcription.

Les quadruplexes

Les G-quartets, quadruplexes de guanines, sont constitués par la configuration de quatre brins d'ADN ou d'ARN comprenant des blocs de plusieurs guanines consécutives où se forment des tétrades ou quadruplets de guanines. Les quatres brins sont généralement orientés parallèlement mais plusieurs conformations sont possibles.

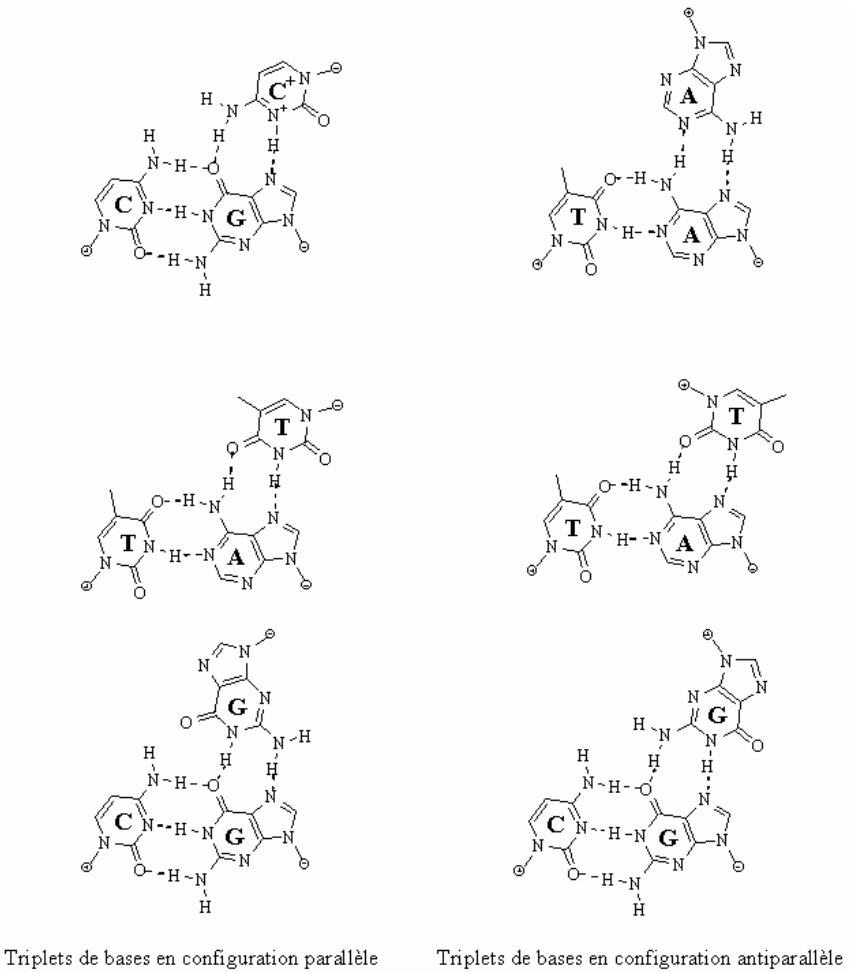


FIG. 1.6 – *Différentes formations de triplets de bases en configuration Hoogsteen à gauche et Hoogsteen inversé à droite. La base du troisième brin d'ADN est celle du dessus et son orientation est symbolisée par un + ou un − (d'après [Gui02]).*

1.1.3 l'ARN

Une molécule d'ARN correspond à la copie complémentaire[5] (Le C est remplacé par le G, le A par le U, le G par le C et le T par le A) de régions particulières de l'ADN appelées gènes. Cette étape de synthèse est appelée transcription et est effectuée par l'ARN polymérase.

1.1.3.1 Composition chimique

L'ARN est un polymère linéaire composé spécifiquement de ribonucléotides (Fig. 1.7) (le désoxyribose de l'ADN est substitué par un ribose). Les bases azotées, constituant la partie variable de la molécule à l'instar de l'ADN, sont l'uracile (remplace la thymine), l'adénine, la

5. Certains virus, appelés rétrovirus, ont leur génome composé par deux molécules d'ARN double brin. Leur matériel génétique est transformé en ADN grâce à une enzyme la *transcriptase inverse*. Cette transformation, la *rétro-transcription*, va permettre au virus de s'intégrer dans l'ADN de la cellule infectée. Le VIH est un rétrovirus de même que le VHA et le VHC.

guanine et la cytosine. L'orientation de la molécule est la même que celle de l'ADN, 5' vers 3' (Fig. 1.3).

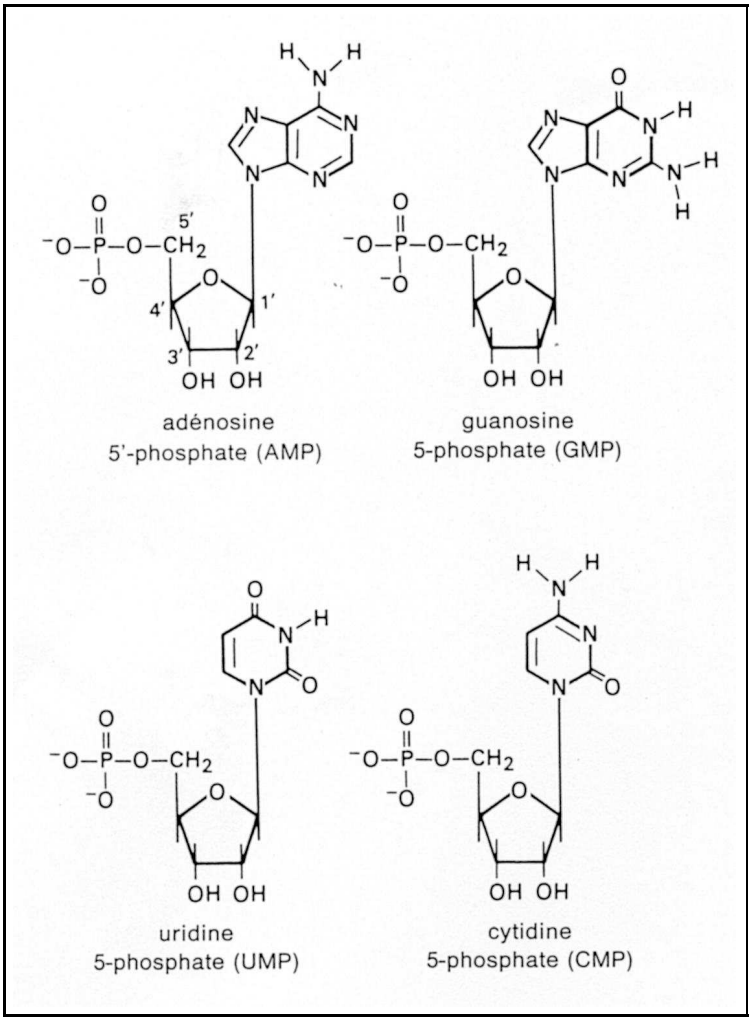


FIG. 1.7 – *Les ribonucléotides de l'ARN (d'après [SB92]).*

1.1.3.2 Structure de l'ARN

La substitution du désoxyribose contenu dans l'ADN par un ribose pour l'ARN provoque des modifications sur la structure de l'ARN:

- elle empêche les duplexes d'ARN d'adopter la conformation B en favorisant la conformation A ;
- elle confère à la molécule une liberté d'interaction ;
- elle lui permet de catalyser des réactions biochimiques.

La stabilité de la molécule est le résultat d'interactions de type hydrophobe entre les bases empilées (identique à l'ADN) ainsi que des liaisons hydrogènes entre deux segments complémentaires.

Contrairement à l'ADN, l'ARN est en général une molécule simple brin, ce qui autorise les bases à s'apparier entre elles par des liaisons hydrogènes. Une molécule d'ARN a ainsi la possibilité de se replier sur elle même ou d'interagir avec d'autres molécules.

Ces appariements intramoléculaires ou intermoléculaires jouent un rôle central dans la fonction de tous les ARN en leur imposant une structure tridimensionnelle unique.

Forces d'appariement

Les liaisons hydrogènes se forment entre les bases capables de partager un proton (atome d'hydrogène). Comme pour l'ADN, chaque base ne peut s'apparier qu'avec une seule autre (purine avec pyrimidine). Cependant, il existe sur les bases d'autres groupements qui peuvent aussi être impliqués dans les liaisons hydrogène et ainsi participer à d'autres appariements.

Certains appariements sont canoniques (U/A et G/C) avec un nombre de liaisons hydrogène pouvant être différent de celui des appariements de type Watson-Crick. D'autres sont non canoniques et forment par exemple des paires A/C et G/A (Fig 1.8). La formation de triplets de bases où une base peut être engagée dans un premier appariement de type Watson-Crick et dans un second appariement de type Hoogsteen avec une troisième base sont aussi possibles. Selon la force des liaisons entre les bases, on peut différencier 4 classes selon un gradient de l'énergie résultante des interactions :

- appariements canoniques ;
- appariements de type Wobble (G-U) : plus faible énergie ;
- appariements de type Hoogsteen : très faible énergie ;
- il existe douze familles dont un grand nombre ne sont pas possibles[6] [LW01] ;

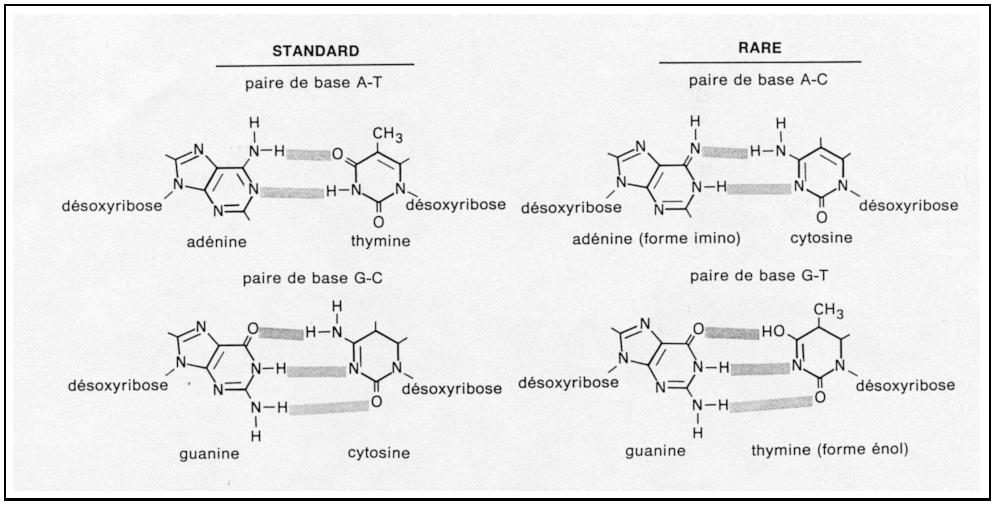


FIG. 1.8 – *Comparaison de paires de bases normales et paires inhabituelles, formées rarement (d'après [SB92]).*

L'énergie libre d'une suite de bases appariées est approchée par la somme des énergies de chaque paire de bases [TUM71]. Cette énergie est mesurée en kilocalorie par mole (kcal/mol).

6. Pour revue : http://prion.bchs.uh.edu/bp_type/bp_structure.html

1.1.3.3 Classification des structures d'ARN

La plupart des ARN fonctionnels adoptent une structure essentielle à leur fonction biologique. Même l'ARNm, souvent considéré comme une simple succession de codons flanquée de séquences spécifiques en 3' et 5', possède une structure importante pour sa fonction biologique (1.9).

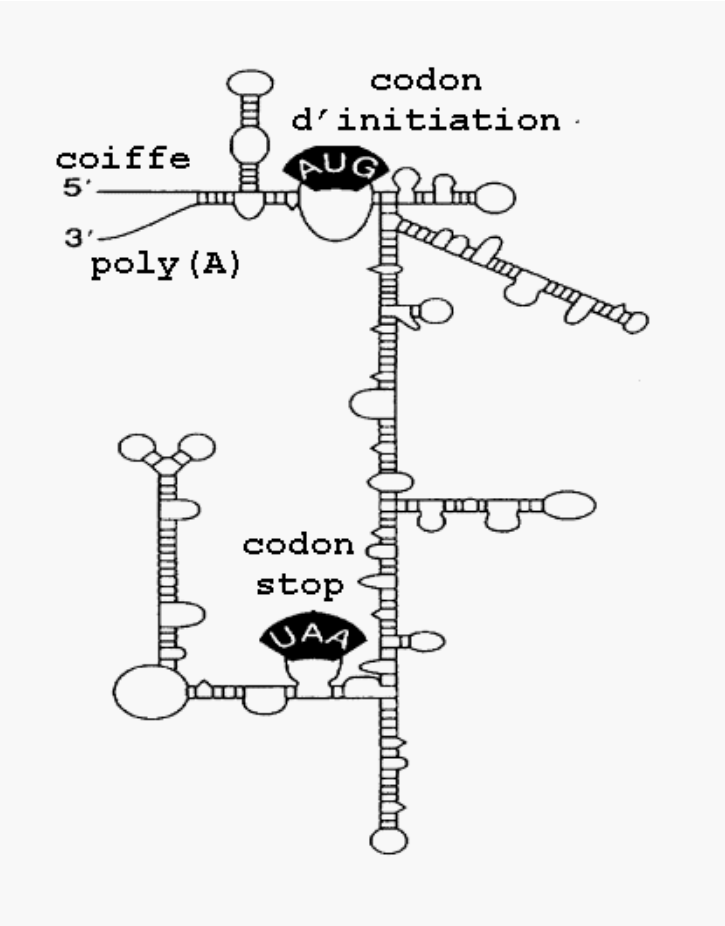


FIG. 1.9 – *L'ARNm adopte naturellement des structures secondaires multiples qui sont en équilibre. Une géométrie particulière est proposée ci-dessus pour l'ARNm de la β-globine (d'après [Bra98]).*

Il existe une classification des ARN selon la hiérarchisation de leurs structures [LW01]. Les structures des ARN peuvent être classées selon une hiérarchie à trois niveaux détaillées dans les sections suivantes [LW01].

La structure primaire

La structure primaire rend compte de la séquence nucléique. Pour une séquence donnée d'ARN, il existe une quantité importante de repliements possibles. La molécule tend à se replier

selon la structure qui minimisera l'énergie libre.

La structure secondaire

Une structure secondaire d'une séquence d'ARN est une représentation partielle de la structure tertiaire. Du point de vue énergétique, sa formation est le résultat de modifications importantes de diverses propriétés physiques de la molécule d'ARN, en spectrométrie d'absorption (densité optique à 260 nm), en dichroïsme circulaire ou en mobilité électrophorétique. La détermination d'une température de fusion de cette structure secondaire, à l'aide des méthodes qui viennent d'être citées, est un élément important pour estimer l'énergie libre de la structuration d'un ARN et définir les conditions ("natives" ou dénaturantes) de son analyse expérimentale. Les liaisons entre bases de l'ARN de type Watson-Crick (c'est-à- dire celles qui impliquent la formation de paires A/U ou G/C) sont dites canoniques, alors qu'un appariement G/U est appelé Wobble. D'autres appariements non-canoniques par liaisons hydrogènes existent tels les paires G/A ou U/U [BDBH99], ou à l'extrémité d'hélices, tels les paires A/A, G/G ou G/A [TS82].

Quand on répartit sur un cercle la séquence d'un ARN et qu'on relie par un trait les paires de nucléotides appariées, une structure secondaire de cet ARN correspond à l'ensemble de tous les appariements représentés par des traits non recoupés par un autre segment (Fig1.10). Sur cette même structure secondaire, tout ajout d'un appariement dont le segment croise au moins un segment est un contact tertiaire.

La structure secondaire d'un ARN est composée par un ensemble d'un ou plusieurs éléments ou motifs élémentaires (Fig. 1.11) (pour revue [CT91]). Le plus connu est le motif en tige-boucle ou "épingle à cheveux" constitué d'une région en hélice terminée par quelques nucléotides non appariés formant une boucle. Le rôle de ces boucles est varié : stabilisation, interaction longue distance dans l'ARN ou interaction ARN-protéine.

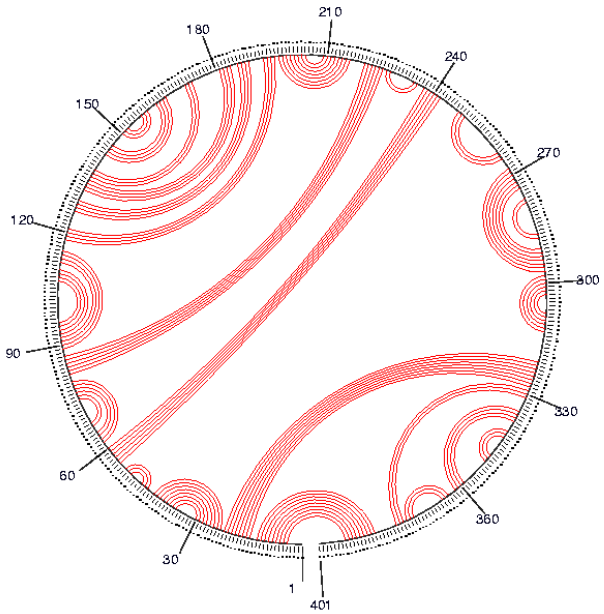


FIG. 1.10 – *Représentation circulaire de la structure d'un ARN, d'après Zuker (URL : http://www.bioinfo.rpi.edu/ zukerm/rna/).*

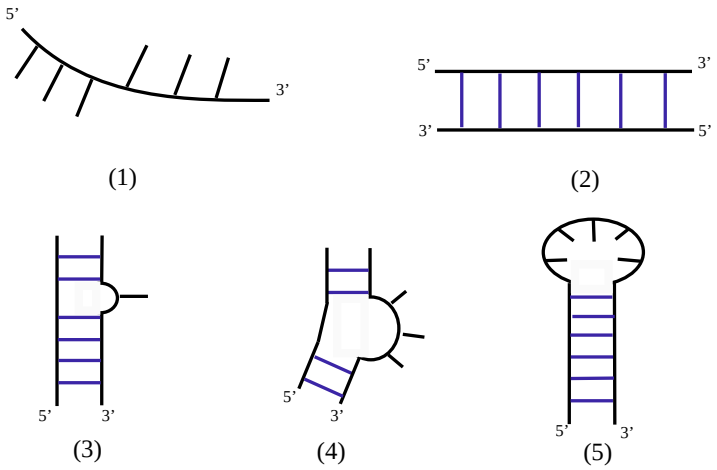


FIG. 1.11 – *Éléments de la structure secondaire des ARN : (1) élément de séquence, (2) région double brin, (3) hélice avec une renflement d’une base, (4) hélice avec un renflement de trois bases, (5) structure en tige-boucle.*

La structure tertiaire

La structure tertiaire représente un niveau de repliement supérieur de l’ARN. Elle est constituée essentiellement de l’interaction de motifs secondaires sur courtes ou longues distances. L’ensemble forme une structure spatiale plus complexe que celle obtenue par la structure secondaire. Un exemple de motif tertiaire est donné par le pseudo-noeud.

Le pseudo-noeud (Fig. 1.12) est constitué par l’interaction de la boucle d’une structure secondaire avec une séquence complémentaire située en dehors de la boucle. Cette séquence ne traversant pas la boucle, la structure a été dénommée pseudo-noeud. Les pseudo-noeuds sont très souvent présents dans les ARN. Ils fournissent souvent un point d’ancrage aux protéines, notamment dans les ARNm et ARNr.

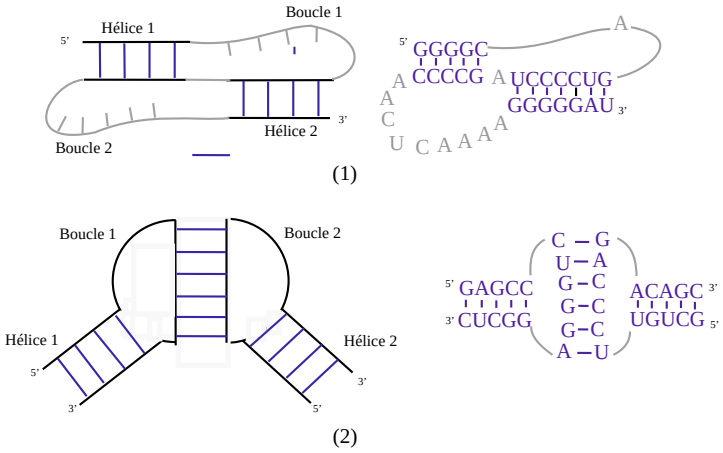


FIG. 1.12 – *Éléments de la structure tertiaires des ARN : (1) pseudo-noeud. (2) “kissing hairpins”.*

1.1.4 Le dogme central - thème et variations

La première section de ce chapitre a présenté le dogme central correspondant à la formulation originale de Francis Crick (ADN $\rightarrow$ ARN $\rightarrow$ protéine). Cette vision du flux d'information dans les êtres vivants impliquait une vue limitée du rôle des ARN.

En effet, selon le dogme central, l'ADN avait deux fonctions de base : la duplication (division cellulaire) et la transcription (synthèse d'ARNm pour ensuite être traduit en protéines). L'ARN, quant à lui, était vu comme l'intermédiaire instable entre l'ADN et les protéines.

La découverte en 1970 d'une enzyme particulière par Temin et Baltimore a bouleversé cette vision des choses en mettant en évidence l'existence d'un flux d'information de l'ARN vers l'ADN. Cette enzyme, appelée la transcriptase inverse, est essentielle pour la reproduction des virus à ARN. Elle permet une synthèse d'ADN double brin (une ADN polymérase) à partir d'une molécule à ARN. Cette découverte a eu également un apport essentiel à la génétique moléculaire en permettant de synthétiser in vitro des molécules d'ADN complémentaires à des molécules d'ARN.

Deux possibilités supplémentaires pour le transfert de l'information chez les virus à ARN

Transfert ARN $\Rightarrow$ ARN : l'objectif de ce transfert d'information est double :

- fournir l'ARN génomique pour la poursuite du cycle infectieux,
- synthétiser l'ARNm codant pour les protéines virales nécessaires au processus infectieux.

Transfert ARN $\Rightarrow$ ADN : cette étape est nécessaire pour permettre l'intégration du génome du virus à ARN dans le génome à ADN de la cellule hôte.

La figure 1.13 illustre la vision moderne du dogme central de la biologie moléculaire où une branche au diagramme initial a été ajoutée pour distinguer les ARN messagers des ARN fonctionnels .

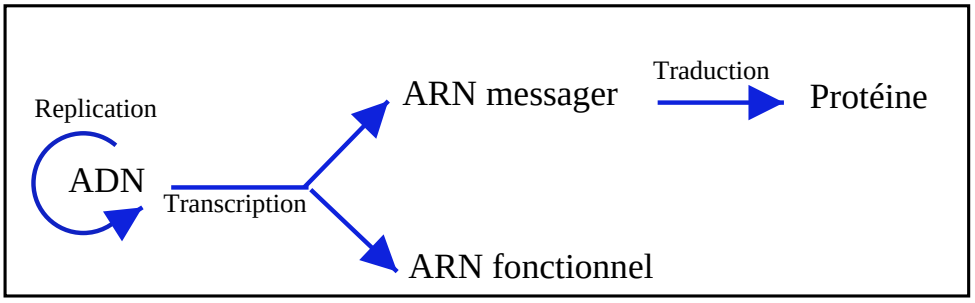


FIG. 1.13 – *Dogme central de la biologie moléculaire moderne*

1.2 Les ARN : une famille en expansion

Au cours de ces dernières années, de nombreux progrès technologiques ont rendu possible le séquençage à grande échelle du génome de plusieurs espèces bactériennes et eucaryotes et entraîné l'augmentation du nombre de séquences génomiques dans les banques de données.

La première étape, pour comprendre le fonctionnement de la cellule, a consisté à décrypter ces données en cherchant à localiser dans les séquences génomiques les régions correspondant à des gènes codant pour des protéines. Le premier résultat surprenant a été d'observer que la fraction codante (nombre de gènes divisé par la taille du génome (Tab 1.2) décroît de façon spectaculaire en fonction de la complexité des organismes. Par exemple, environ 10% de régions chez les archae-bactéries et bactéries sont non-codantes, 30% chez la levure (eucaryote unicellulaire), 70% chez le ver, l'arabette et enfin 98 % chez l'homme.

Ces résultats sont surprenants dans la mesure où on s'attendait à trouver un nombre plus important de gènes. Les phénomènes tels que "l'épissage alternatif " et les modifications post-traductionnelles différentielles, permettant de générer plusieurs protéines à partir d'un seul ARN, ne suffisent pas à expliquer ce paradoxe.

Aussi, l'intérêt manifesté vis à vis des régions non codantes est de plus en plus important, surtout depuis les découvertes de petits ARN fonctionnels impliqués dans des rôles variés du fonctionnement cellulaire. Aujourd'hui, le rôle des petits ARN dans les réseaux de régulation et autres processus est évident, aussi bien chez les procaryotes que chez les eucaryotes.

Espèces	Taille (Mb)	Nombre s de gènes	Fraction codante
Archae-bactéries			
Pyrococcus abyssi	1,76	2,065	91%
Pyrococcus horikoshii	1,73	2,064	91%
Bactéries			
Bacillus subtilis	4,2	4,100	97%
Esherichia coli	4,6	4,289	88%
Eucaryotes			
Saccharomyces cerevisiae	13	6400	68%
Arabidopsis thaliana	125	25500	29%
Caenorhabditis elegans	100	18000	27%
Drosophila melanogaster	180	13 600	13%
Homo sapiens	3.000	24 500	1,4%

TAB. 1.1 – *Proportion des régions codantes dans des génomes des trois taxons.*

1.2.1 L'état actuel des connaissances chez *E. coli*

En 2001, trois groupes ont effectué des recherches dans les régions intergéniques de *E. coli* et ont découvert 31 nouveaux petits ARN [AHV+01, RKJE01, WRR+01] (les approches utilisées sont abordées dans la section 1.4.3.3). Une récente compilation situe le nombre de gènes non codants à 55 et évalue à plus de 900 le nombre de candidats non redondants dont l'existence n'a pas encore été démontrée [HAM03]. La plupart des recherches effectuées sur *E. coli* se sont spécifiquement intéressées à des ARNnc codés par des gènes indépendants [AHV+01, WRR+01, CLH+02], alors que les ARNnc peuvent être générés par de longs transcrits initiaux comme dans le cas du 6S chez *E. coli* [HZWF85]. Ainsi, Vogel *et al* ont récemment élevé le nombre potentiel de candidats chez *E. coli* à 62 [VBT+03] en identifiant des petits ARN co-transcrits avec des ARN messagers (pour revues sur les fonctions [GWa03, WRR+01]).

1.2.2 Nomenclature des gènes d'ARN

Les terminologies trouvées dans la littérature varient selon les auteurs :

sARN : petit ARN (small RNA)
ncARN : ARN non codant (non-coding RNA)
fARN : ARN fonctionnel (Functional RNA)

Pour plus de clarté, nous désignerons par la suite les ARN non traduits en protéine par le terme d'ARN non codants (ARNnc).

1.2.3 Diversité des ARN non codants

Historiquement, les premiers ARN non codants identifiés furent les ARN de transfert et les ARN ribosomaux, présents dans toutes les espèces et directement impliqués dans la traduction des protéines. L'étude des ARN non codants a réellement démarré dans les années 80 avec la découverte des propriétés catalytiques de certains ARN[7].

Ils exercent des fonctions variées à des niveaux très divers (pour revue [Sto02]): activité catalytique, épissage des ARN messagers, maturation des ARN ribosomiques, transcription, traduction, transport cellulaire, édition des ARN, stabilité des ARN messagers, dégradation cellulaire et certainement d'autres rôles qui n'ont pas encore été mis en évidence (Fig. 1.14).

7. Les protéines ont été longtemps considérées comme étant les seules biomolécules douées d'activité enzymatique. Cette certitude a été remise en cause par la découverte des propriétés catalytiques de l'ARN par Sidney Altman et Thomas R. Cech en 1989 (prix Nobel de chimie 1989 [Nor89]). Les ARN possédant une activité enzymatique sont désignés par le terme ribozyme.

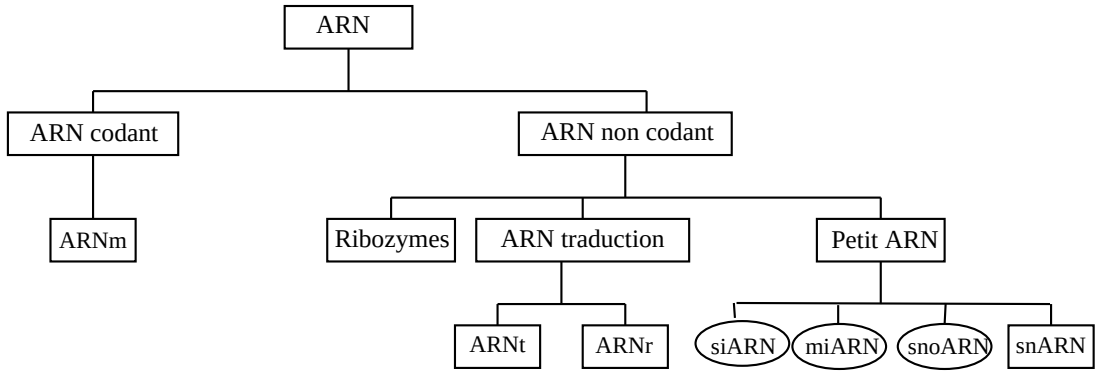


FIG. 1.14 – *Familles des ARN. La section 1.3 décrit les familles représentées dans les cercles.*

L'intérêt actuel de la communauté scientifique se porte sur leur localisation, leur compréhension, leur mode d'action et leur rôle.

1.2.4 Le mode d'action des ARN non codants

Les connaissances actuelles sur les ARN non codants montrent déjà l'énorme potentiel de ces molécules. Les molécules d'ARN jouent un rôle crucial dans une large gamme de processus cellulaires essentiels [Che99, Sto02] à travers leur capacité à adopter un repliement tridimensionnel bien précis, à réarranger leur conformation et à reconnaître très spécifiquement une grande diversité de ligands (protéines ou acides nucléiques). De plus, le repliement tridimensionnel d'une molécule d'ARN n'est pas statique, la molécule ayant à tout moment la capacité de subir des réarrangements conformationnels modulés par l'interaction de nombreux ligands. Le tableau 1.2 citent quelques exemples d'ARN non codants regroupés en fonction du ligand avec lequel ils interagissent.

Le contexte biologique de notre travail se porte plus particulièrement sur les ARN non codants dont le mode d'action implique une interaction intermoléculaire avec un à plusieurs ARN. Les sections suivantes se proposent d'illustrer leur diversité au travers de quelques exemples (les exemples décrits sont soulignés dans le tableau 1.2).

Fonction cellulaire	ARN non codants	Taille en nucléotides	Mécanismes
		Cible : ADN	
Réplication	ARN télomérique	400 à 500	Synthèse des télomères [Bla00]
Extinction de gènes	ARN Xist	17000	Inactivation du chromosome X [KK00]
		Cible : ARN	
Maturation des ARN	ARNase P	370	Clivage de l'extrémité 5' des ARNt
	snARN	100 à 300	Épissage des introns des pré-ARNm [Ree00]
	snoARN C/D et H/ACA	70 à 350	Clivages des pré-ARNr [BMN+95, BSF96]& modifications de nucléotides des ARN
	gARN	80	Edition de l'ARN [STS+00]
	miARN/siARN	21 à 25	Dégradation de l'ARNm [DPS03]
Traduction	miARN	21 à 25	Inhibition de la traduction [RSB+00]
	DsrA	90	Stimule la traduction [SGG96]
	OxyS	100	Inhibition de la traduction [AWFZ+97]
Virulence	ARNIII	517	Activation de la traduction [MTvGA95]
Transposition	ARN-OUT	69	Régulation [KSLK89]
		Cible : Protéines	
Traduction	ARNr	1500 à 5000	Décodage de l'information génétique
	ARNt	70 à 90	
Stabilité	ARNtm	350	Dégradation des protéines [GH94]
Trafic intracellulaire	ARN SRP	114	Translocation des protéines [Que94]
Transcription	ARN 6S	21 à 25	Régulation de la transcription [WS00]

TAB. 1.2 – *Catalogue non exhaustif d'ARN non codants interagissant avec des protéines (d'après [CB03]).*

1.3 Quelques exemples de familles d'ARN non codants

1.3.1 Les ARN impliqués dans la maturation d'autres ARN

La classe d'ARN non codants la plus importante en terme de nombre, après les ARNr et ARNt, est celle des snoARN (*Small Nucleolar RNAs*). Ces petits ARN nucléolaires non codants (pour revues : [BC97, CB98, MF95, SS97, TK97, Kis02, BCH02, GFB+02]) ont été mis en évidence grâce aux modifications qu'ils guident et induisent sur d'autres ARN [KLHB+96].

1.3.1.1 Caractéristiques et fonctions

On distingue deux familles de snoARN sur la base de la présence d'éléments caractéristiques : les antisens à boîtes C/D et les antisens à boîtes H/ACA.

Les snoARN à boîtes C/D (Fig A 1.15) contiennent les deux courts motifs de séquence conservés dits boîte C (5'-R[8]UGAUGA-3') et boîte D (5'-CUGA-3') à quelques nucléotides seulement des extrémités 5' et 3', respectivement. Une copie dégénérée de chacun de ces motifs (C' & D') peut également être présente dans la région centrale, selon les espèces. La quasi-totalité de ces snoARNs contiennent de longues complémentarités (de 10 à 21 nt) à différentes régions conservées d'autres ARN (ARNr, ARNt, snARN, ARNm) et ils ont donc été appelés snoARNs antisens à boîtes C/D [BMN+95]. Généralement les deux boîtes C et D font partie d'une structure terminale caractéristique (Fig (A) 1.15) dans laquelle les nucléotides 5'- et 3'-terminaux sont appariés. L'intégrité de cette structure terminale est essentielle pour la maturation correcte du snoARN intronique (chez les eucaryotes) et sa stabilité dans la cellule [CB96]. Les deux boîtes C et D sont impliquées dans l'association de tous ces snoARNs avec la fibrillarine, une protéine nucléolaire requise pour la maturation normale du pré-ARNr et pour sa méthylation [MF95]. Pour les premiers snoARNs antisens à boîtes C/D qui avaient été identifiés, la caractérisation d'homologues dans des espèces très distantes avait permis d'observer une conservation particulièrement forte de la séquence de complémentarité à l'ARNr au cours de l'évolution, suggérant que l'appariement de ces snoARNs au pré-ARNr jouait un rôle essentiel dans la fonction de ces snoARNs [QNM+94].

La deuxième famille, dite H/ACA, a été identifiée plus tard. Elle est définie par la présence systématique du tri-nucléotide conservé H/ACA à 3 nucléotides de l'extrémité 3' [BSF96] et la présence d'un ou plusieurs longs domaines indépendants de structure secondaire organisés en tige-boucle. Lorsque plus d'un domaine existe, ces deux domaines sont séparées par une région charnière simple brin riche en purine et incluant une boîte H (5'-ANANNA-3') (Fig (B) 1.15). Le triplet ACA, situé à l'extrémité 3', est requis pour la maturation et la stabilité de ces snoARNs [BSF96]. Chez la levure, les snoARNs de la famille ACA co-précipitent tous avec la protéine nucléolaire, `Gar1` [GLCF+92, BSF96, GBK97]. Ces snoARN de type H/ACA possèdent aussi des complémentarités à d'autres ARN (essentiellement ARNr). Cependant, les complémentarités à l'ARN cible sont organisées différemment (Fig (B) 1.15).

8. R dans le code IUPAC 2.1 signifie une purine.

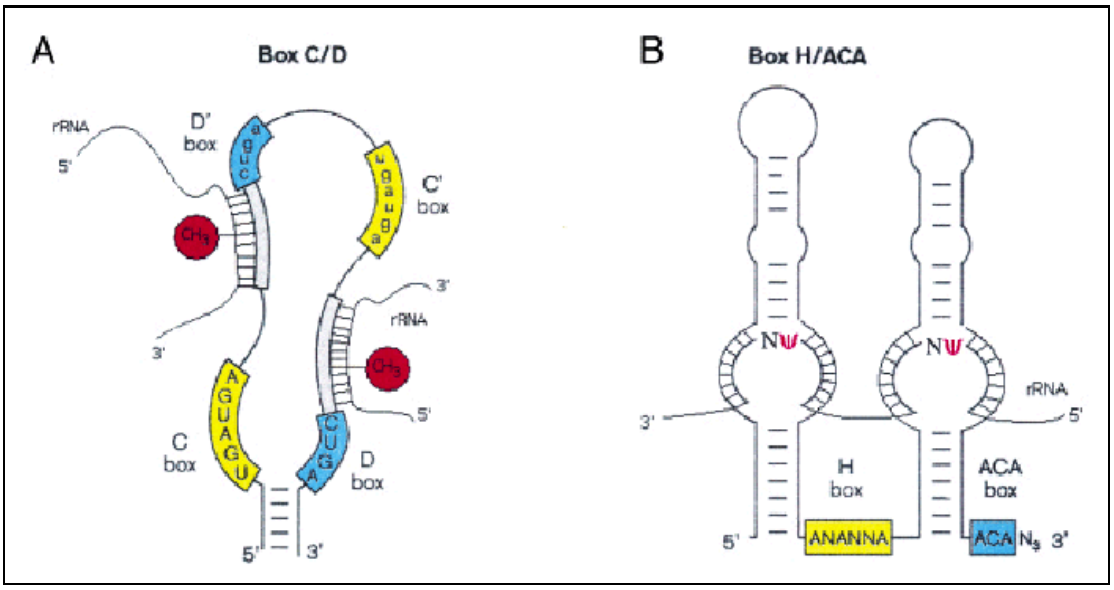


FIG. 1.15 – *(A) : snoARN à boîtes C/D. (B) snoARN à boîtes H/ACA (d'après [DOL01]).*

Ces deux familles d'ARN nucléolaires induisent respectivement sur leurs cibles ARN une méthylation en 2'-O sur le ribose ou une pseudouridylation (isomérisation de l'uridine en pseudo uridine). Ces modifications post-transcriptionnelles sont précisément localisées (en général sur des positions hautement conservées) grâce à la formation d'une structure particulière en duplex snoARN/ARN cible. Les structures secondaires formées par les snoARN et leur cibles jouent donc un rôle prépondérant dans les phénomène d'appariement-modification.

Les snoARN sont présents dans de nombreux organismes (vertébrés, plantes, levures, archaebactéries [GCEB00, OLR+00], ...). Ils ont pour cibles les ARNr, les ARNt, les snARN et certainement d'autre types d'ARN pour lesquels aucune démonstration n'est aujourd'hui disponible. Les gènes de snoARN se retrouvent majoritairement dans les introns d'un gène hôte (vertébrés), parfois en unités géniques autonomes (plantes, levure, archaebactéries) ou parfois même dans certains exons (plantes). De plus, de nouveaux snoARN, dépourvus de complémentarité pour les ARNr et snARN et qui ne sont pas exprimés ubiquitairement (localisés dans les introns de gènes à expression tissu-spécifique) ont été mis en évidence, ce qui ouvre des perspectives quant à l'étendue des fonctions des snoARN et justifie leur étude approfondie.

1.3.1.2 Une nouvelle fonction pour les snoARN

Cavaillé *et al* ont identifié [CBK+00] une nouvelle classe de petits ARN nucléolaires exclusivement exprimés dans le cerveau des mammifères. Les snoARN mis en évidence interviennent pour contrôler l'expression de gènes spécifiquement exprimés dans le cerveau, en modifiant sélectivement la structure de certains ARN messagers. Ces travaux montrent également que ces snoARN sont absents chez des individus atteints du syndrome de Prader-Willi (l'une des 10 maladies génétiques les plus fréquentes), laissant ouverte la possibilité que des défauts d'expression de ces snoARN puissent être impliqués dans l'étiologie de cette maladie.

1.3.2 Les ARN impliqués dans la régulation de fonctions

La cellule est capable d'adapter rapidement son métabolisme aux modifications de l'environnement grâce aux mécanismes de régulation qui permettent de contrôler la production de protéines particulières. La régulation de l'expression des gènes se fait à plusieurs niveaux selon les cas. Par exemple, lors de l'initiation de la transcription (activation ou non du promoteur), au niveau post-transcriptionnel (jouant sur la stabilité ou sur le contrôle des étapes de maturation des ARN messagers), ou bien au niveau post-traductionnel en agissant sur la vitesse de traduction ou sur la stabilité de la protéine. Selon l'effet produit par la régulation d'une molécule, on parle de molécule effectrice (activation) ou inhibitrice (inhibition).

Les régulateurs au niveau transcriptionnel ou post-transcriptionnel permettent d'adapter rapidement la croissance bactérienne aux conditions du milieu extérieur et à divers stress. En effet, une régulation effectuée dans les premières étapes de la synthèse des protéines présente l'avantage d'être efficace et plus économique pour la cellule.

Les régulateurs peuvent être des molécules protéiques ou d'ARN. Pour qu'une régulation soit effectuée, il est nécessaire qu'une interaction entre la molécule régulatrice et la molécule à réguler soit possible. Les ARN régulateurs ou riborégulateurs ont ainsi la capacité de pouvoir interagir avec des molécules d'ADN, d'ARN ou protéiques. Dans les deux premiers cas, les interactions de type ADN/ARNm ou ARN/ARNm impliquent deux régions présentant une complémentarité de séquence stricte ou partielle.

La régulation d'une protéine par un ARN est généralement inhibitrice et la molécule d'ARN, en interagissant avec la protéine, la séquestre. D'autres ARN régulateurs sont multi-fonctionnels et contiennent à la fois des sites de reconnaissance pour des protéines et des séquences ciblant des ARNm (pour revue détaillée [Kol01]).

1.3.2.1 Avantages des riborégulateurs

L'utilisation de régulateurs de type ARN par la cellule n'est peut-être pas due au hasard. Le premier argument en faveur de cette hypothèse est l'économie d'énergie et le gain de vitesse effectués par la cellule lors de la synthèse du riborégulateur. En effet, la synthèse d'une protéine régulatrice nécessite la synthèse intermédiaire d'un ARN messager augmentant ainsi le nombre d'étapes et d'éléments nécessaires. De même, la vitesse de dégradation d'un ARN est relativement rapide et permet ainsi à la cellule de moduler précisément la fonction biologique de ces molécules en réponse à des signaux divers. Enfin, les ARN antisens ont la capacité d'interagir rapidement allant dans le sens de l'efficacité du contrôle qui semble étroitement liée à la vitesse d'interaction. L'utilisation de riborégulateurs permet donc à la cellule de contrôler une fonction biologique dans un intervalle optimal.

1.3.2.2 Mode d'action

Les riborégulateurs se regroupent dans trois grandes classes [Kol01] en fonction de la molécule avec laquelle ils interagissent:

Dans la première classe, le riborégulateur contrôle indirectement la synthèse de la protéine ciblée en interagissant avec une protéine de régulation de la transcription ou de la traduction (la protéine interagit avec l'ADN ou l'ARN messager). Les ARN de la seconde classe agissent

directement sur la protéine en réprimant son action. La dernière classe d'ARN regroupe les régulateurs de l'expression de protéines interagissant avec l'ARN messager initial (formation d'un duplex ARN-ARN) et on distingue dans ce groupe trois types d'action :

- inhibition de la traduction : la formation du duplex ARN-ARNm au niveau du site de fixation du ribosome sur l'ARNm (RBS : *Ribosome Binding Site*) empêche la fixation du ribosome.
- activation traductionnelle : le RBS d'un ARN messager est impliqué dans une structure en tige-boucle et rend impossible la fixation du ribosome. La fixation d'un petit ARN en amont du RBS empêche la formation de la tige-boucle et permet ainsi la fixation du ribosome.
- dégradation favorisée par un ARN antisens. La fixation de l'ARN sur l'ARN messager crée un site de clivage par une ARNase spécifique des régions bicaténaires.

Les ARN à domaines fonctionnels multiples peuvent agir par différentes combinaisons des mécanismes ci-dessus.

1.3.3 L'interférence ARN : une révolution en génomique fonctionnelle

L'interférence ARN [Han02, Sha01, Han02, PH03] est une découverte importante de ces dernières années en génomique fonctionnelle. Ce processus conservé au cours de l'évolution existe dans une grande variété d'organismes, tel que les protozoaires, les champignons, les plantes et les animaux.

Ce phénomène a été découvert en 1990 par le professeur Jorgensen qui, en essayant de rendre des pétunias plus violets, a obtenu des pétunias blancs. Fortuitement, en essayant d'accentuer la production d'un gène par introduction de ce dernier dans la cellule, il l'a rendu silencieux. Ce phénomène naturel a plusieurs terminologies, il est appelé *post-transcriptional gene silencing* (PTGS) chez les plantes [Bau96], *quelling* chez les levures [RM92] et l'interférence à l'ARN (*RNA interference*) chez les animaux [FXM$^+$98].

Pour comprendre le rôle de l'interférence à ARN dans les cellules, plusieurs études ont été menées sur ce mécanisme. Les résultats obtenus, inhibition de la transposition chez *C. elegans* et modification de la résistance des plantes aux virus [MMK01, MBE$^+$00, KHvLP99], laisse penser que l'interférence ARN serait un mécanisme de défense des cellules eucaryotes.

Les sections suivantes décrivent quelques familles d'ARN et les exemples ont été choisis du fait de leur interaction avec d'autres ARN.

Mode d'action

Les molécules d'ARN double-brin (ARNdb), produites de façon endogène (les virus contenant un génome ARN double brin) ou introduites de façon exogène, induisent la dégradation spécifique de l'ARN messager (ARNm) homologue, inhibant ainsi l'expression des protéines correspondantes.

L'inhibition de la traduction induite par les ARNdb a d'abord été mise en évidence à partir d'études effectuées sur le nématode *C. elegans* [FXM$^+$98, GK95]. Des analyses ultérieures sur la mouche du vinaigre *D. melanogaster* ont permis de mieux comprendre le mécanisme [ELT01] qui nécessite deux étapes décrites ci-dessous :

- Dans un premier temps, de longs ARNdb sont coupés en fragments de 21-23 nucléotides par une enzyme appelée Dicer. Son activité a été identifiée chez *D. melanogaster*

[BCHH01], *C. elegans* [KFB+01] et *H. sapiens* [PDD+02]. Le produit de ce clivage est appelé *small interfering RNAs* (siARN [ELT01, ZTSB00]).

– Les siARN, correspondants aux ARN doubles brins inducteurs, s'associent à un complexe ribonucléasique (`RISC`, pour *RNA Induced Silencing Complex*) lequel intervient à son tour dans la destruction de l'ARNm cible. Les siARN semblent servir de séquence-guide pour reconnaître les ARNm spécifiques [HBBH00].

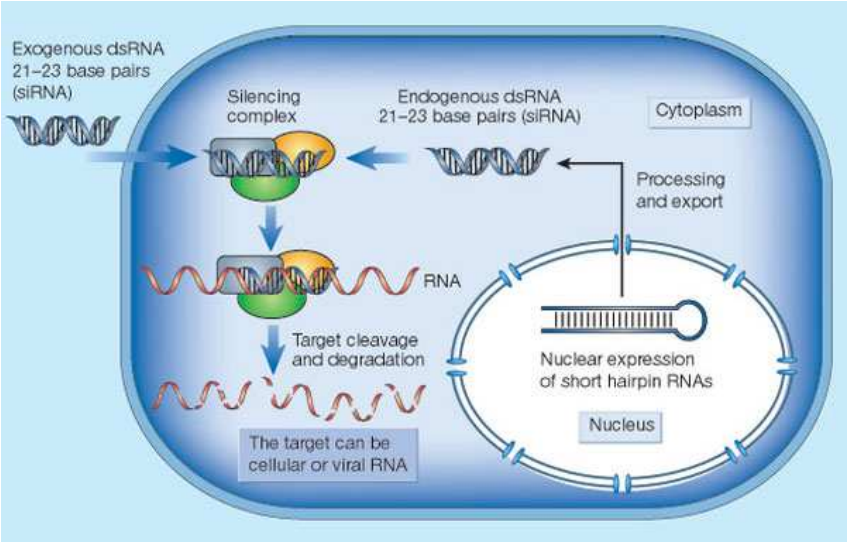


FIG. 1.16 – *Mode de fonctionnement de l'ARN interférence utilisé comme outil thérapeutique sur des cellules humaines. Ce mécanisme cellulaire, responsable de la dégradation de molécules de type ARN, peut être induit, à l'aide de petites molécules synthétiques, les siARN (d'après [Car02]).*

L'ARNi : une nouvelle option thérapeutique

Le mécanisme de l'interférence ARN a ouvert de nouvelles perspectives dans le domaine de la santé. Ce mécanisme cellulaire (Fig 1.16) est ainsi induit artificiellement à l'aide de petites molécules synthétiques, les siARN, et permet l'inactivation fonctionnelle spécifique d'un gène cible avec deux applications majeures dans la recherche pharmaceutique :

– il permet d'étudier indirectement les fonctions des gènes. En sélectionnant les gènes que l'on veut éteindre, il devient possible d'observer les caractères dont ils sont responsables.

– des essais d'immunisation contre le virus du `SIDA` [NMD+02, JTS02] et de la poliomyélite [GKA02] ont été réussis sur des cellules humaines en culture et contre des hépatites virales chez la souris [SLW+03].

  Les retombées d'une telle technique sont innombrables. Ainsi, les petits ARNi seraient eux-mêmes des médicaments et au lieu de bloquer une protéine particulière, comme le font les médicaments classiques, l'interférence de l'ARN empêcherait la production de la protéine.

1.3.3.1 Un exemple de riborégulateur antisens : les microARN

Les microARN (miARN) sont un autre exemple de riborégulateurs interagissant avec une seconde molécule d'ARN. Ce sont des ARN simple brin d'une taille d'environ 22 nucléotides qui sont impliqués dans le contrôle post-trancriptionnel de l'expression de gènes eucaryotes.

Plusieurs miARN ont été identifiés chez le nématode [LA01, LLWB01, LQRLT01, GAH+03, LLW+03], la drosophile [Amb01], les plantes [RRL+02, RWR+02], les vertébrés [LGY+03] et la souris [LQRLT01]... De plus récentes découvertes ont démontré leur influence sur une variété de processus tels que le développement larvaire [RSB+00], la prolifération, l'apoptose cellulaire [BHSR03] et le métabolisme des acides gras [XVGH03]. Les miARN ont la capacité d'inhiber la traduction de protéines par l'inactivation ou la dégradation des ARN messagers correspondants.

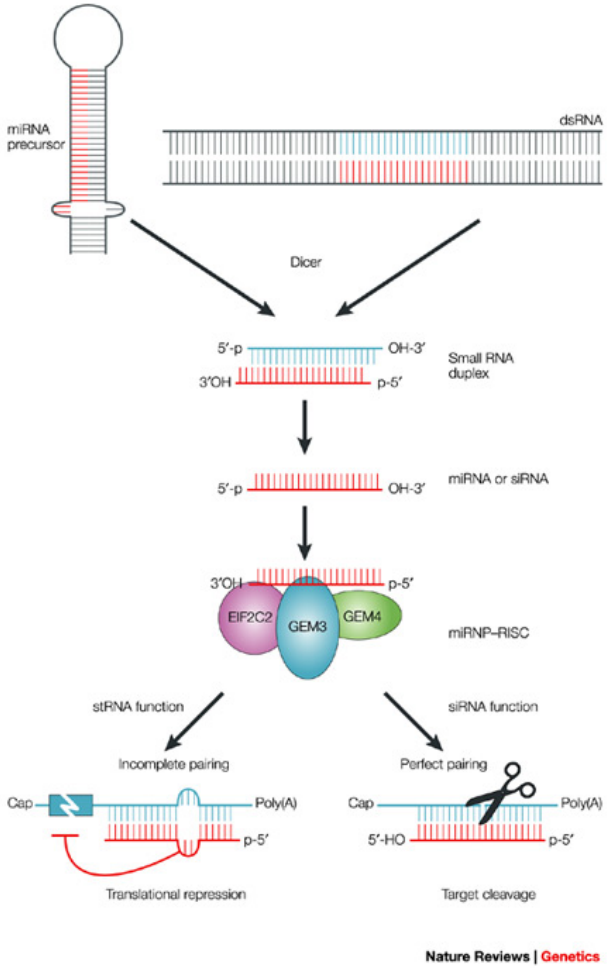


FIG. 1.17 – *La voie de l'interférence par ARN et celle des miARN se recoupent. Dans chacune de ces voies, la protéine* `Dicer` *agit en générant le petit ARN régulateur actif: (1) le stARN ou miARN à partir d'un précurseur en tige-boucle, dont le rôle est de réprimer la traduction de l'ARN messager cible avec lequel il s'apparie de manière imparfaite (2) le siARN à partir de l'ARN double brin, lequel déclenche la destruction d'un ARN cible parfaitement complémentaire (d'après [Buc03]).*

Les deux représentants de cette famille dont la fonction et le rôle ont été le mieux compris sont les ARN `let-7` (Fig. 1.18) et `lin-4` [RSB+00, Amb01, LFA93, WHR93], impliqués dans la régulation d'une famille d'ARN messagers jouant un rôle dans le développement larvaire chez *C. elegans*.

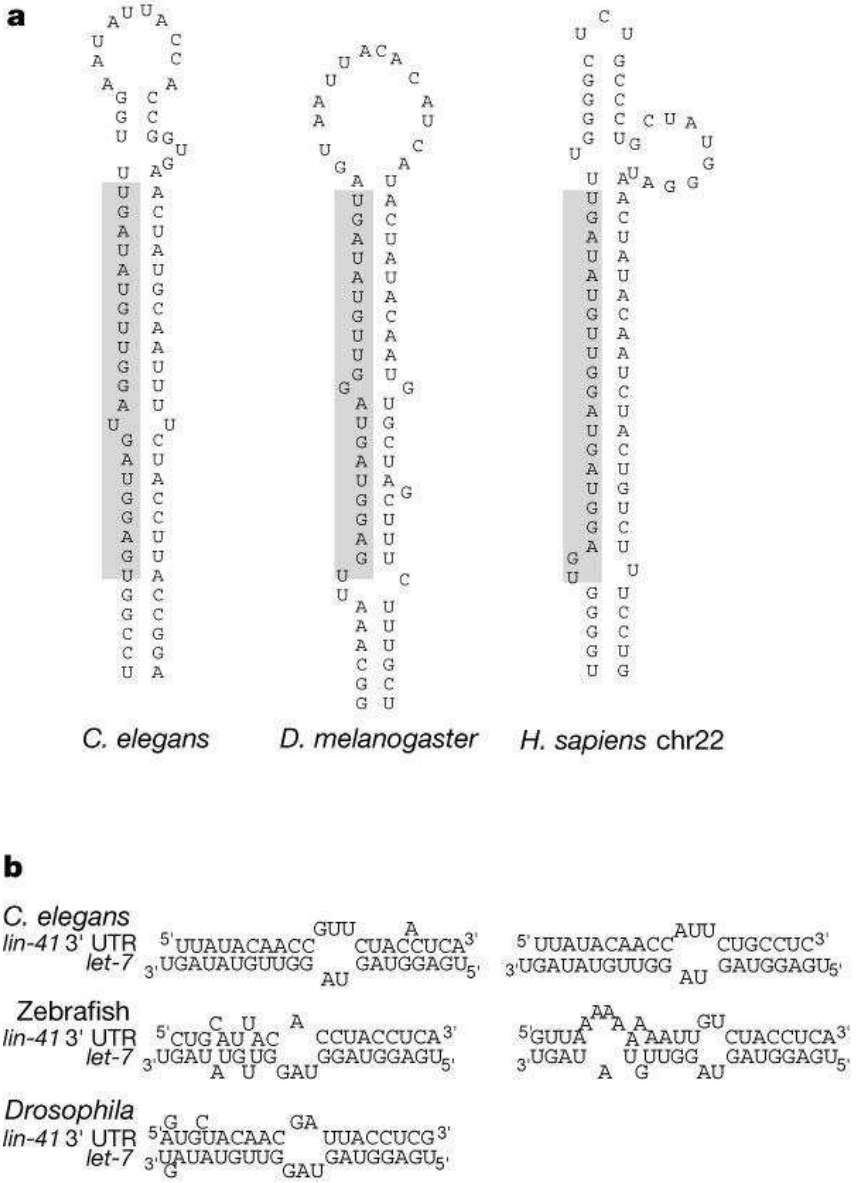


FIG. 1.18 – **a :** *Structure secondaire des ARN précurseurs de* `let-7` *chez trois organismes. La région de 21 nucléotides correspondant aux ARN* `let-7` *matures dans les organismes est grisée.* **b :** *les ARN messagers* `lin-41` *des trois organismes contiennent des sites complémentaires de* `let-7`*, (d'après [PRS+00]).*

Ils régulent la synthèse protéique en se liant de manière imparfaite à une région non traduite de l'ARN messager cible (Fig. 1.18). La maturation des gènes `let-7` et `lin-4` à partir d'un précurseur en tige-boucle de 70 paires de bases relève d'un mécanisme similaire à l'interférence

ARN. Les miARN et le phénomène d'interférence à ARN présentent ainsi des similitudes. Ils utilisent tous les deux la protéine `Dicer` (Fig. 1.17).

De plus il a été récemment prouvé [DPS03] que les miARN présentant une complémentarité parfaite avec un ARN messager s'associent parfois au complexe `RISC` et induisent la dégradation sélective de cet ARN messager, comme le font les ARN interférents. Le phénomène où les ARN interférents inhibent de façon transitoire la traduction d'un ARN messager a aussi été observé pour des ARN interférant se liant de façon imparfaite avec des ARN messagers (Fig. 1.18).

Ainsi, la qualité de l'appariement entre deux ARN, la longueur de cet appariement et probablement sa localisation sur la séquence cible pourraient jouer un rôle prépondérant pour le type de régulation sollicitée.

1.3.3.2 Un exemple de riborégulateur antisens codé par un élément génétique accessoire

La régulation de la transposition de l'élément `IS10` est un exemple intéressant pour illustrer le rôle important joué par la structure de l'ARN antisens pour lui assurer sa fonction. Cet ARN régulateur présente les caractéristiques communes à de nombreux ARN antisens, lesquelles lui garantissent une efficacité d'interaction avec son ARN cible.

L'élément transposable `IS10` chez *E. coli* code pour deux ARN (Fig. 1.19) qui sont :

- `ARN-IN` ou ARN messager `tnp` transcrit à partir du promoteur faible P_{in} et codant pour la transposase, responsable de l'étape de transposition.
- `ARN-OUT` transcrit à partir du promoteur fort P_{out}, permettant la régulation de la transposition en interférant avec l'ARN précédent.

Les deux ARN peuvent former ensemble un appariement de 35 paires de bases masquant ainsi le site de fixation du ribosome de l'ARN messager[9] `tnp`.

Le mécanisme de transposition[10] est inhibé par la fixation de l'`ARN-OUT` sur l'`ARN-IN`, lorsque plus d'une copie du transposon est présente par cellule. La traduction de la transposase est réprimée par l'`ARN-OUT` qui, en interagissant avec l'`ARN-IN`, empêche le ribosome de se fixer sur le site promoteur de ce dernier. L'efficacité de cette régulation est étroitement liée à la vitesse d'interaction entre les deux ARN.

La régulation s'effectue en deux étapes (Fig. 1.20) : l'interaction est initiée par l'association d'un petit nombre de nucléotides de la boucle de l'`ARN-OUT` et les nucléotides 5' terminaux de l'`ARN-IN`, formant trois paires G/C. Cette interaction se propage ensuite entre les deux ARN afin de former un duplex étendu en dissociant la structure secondaire initiale de l'`ARN-OUT` [KSLK89].

9. La fixation d'un complexe ribosomique sur l'ARN messager constitue la première étape pour traduire ce dernier en protéine

10. Une transposition consiste en l'incorporation d'acides nucléiques synthétisés hors du génome.

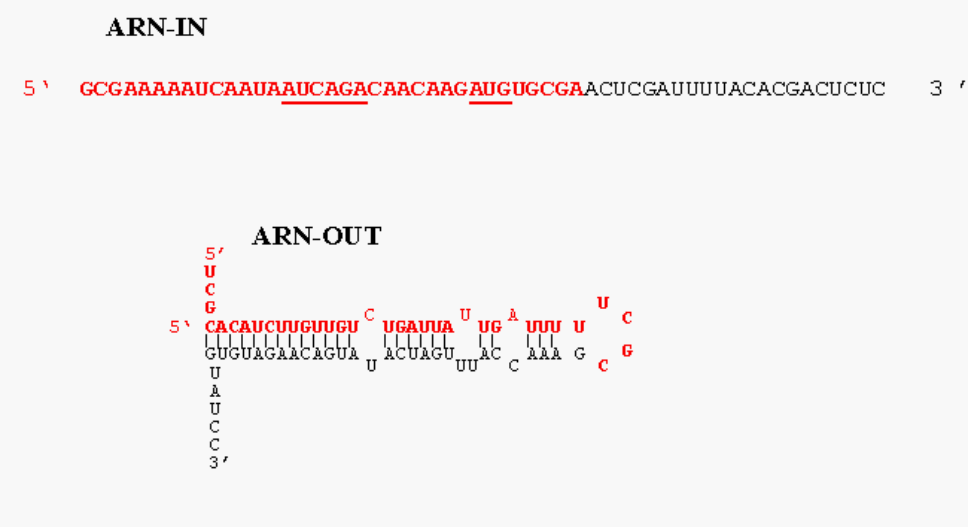


FIG. 1.19 – *Séquence de l'*ARN-IN *et structure secondaire individuelle de l'*ARN-OUT*. Les régions impliquées dans l'appariement des deux ARN sont en gras, et les sites impliqués dans l'étape de traduction de l'*ARN-IN *sont soulignées (seront masquées par l'association des deux ARN) (d'après [KSLK89]).*

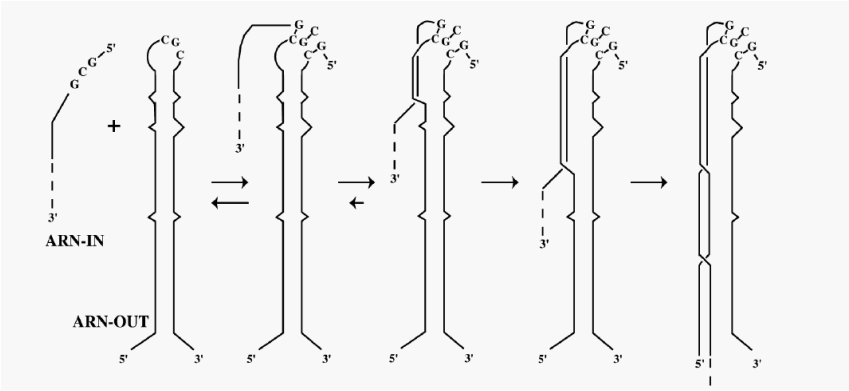


FIG. 1.20 – *Modèle du mécanisme d'appariement entre* ARN-IN *et* ARN-OUT *(d'après [KSLK89]).*

1.3.3.3 Un exemple de riborégulateur multi-fonctionnel

La plupart des ARN régulateurs identifiés n'agissent que sur une cible unique. Cependant, quelques exemples ont été décrits où un seul et même ARN peut agir de façon pleiotropique[11] en contrôlant l'expression de plusieurs gènes. L' ARNIII de *Staphylococcus aureus* [KEER02, BKR+00] constitue un exemple parmi quelques autres.

Les staphylocoques sont des bactéries pathogènes dont l'effet infectieux est lié à la synthèse de nombreuses protéines. L'expression de ces facteurs de virulence est régulée au cours de la croissance bactérienne et au cours des différentes phases de l'infection. Cette régulation est sous

11. Gène pleiotropique: gène qui a plusieurs effets variés sur le phénotype.

la dépendance d'un régulateur global dénommé `agr` (*accessory gene regulator*), lequel contrôle indirectement la transcription d'un autre ARN régulateur : l'`ARNIII`.

L'`ARNIII` réprime l'expression des facteurs d'adhésion et des protéines associées à la paroi, synthétisées en phase précoce de croissance bactérienne, et contrôle positivement au cours de la phase de croissance l'expression des protéines qui seront excrétées par la cellule (Fig. 1.21).

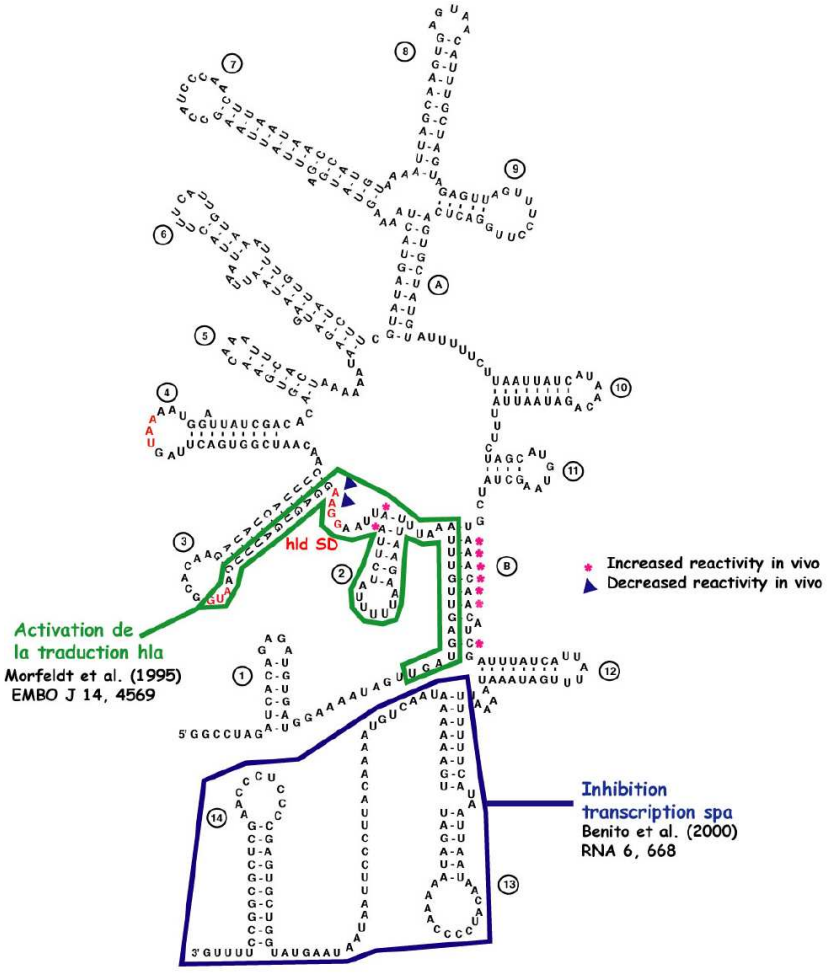


FIG. 1.21 – *Structure secondaire de l'*`ARNIII` *de Staphylococcus aureus. Un premier domaine induit la répression de la transcription du gène* `spa` *codant pour la protéine* `A`*. Un second domaine, complémentaire à l'ARNm* `hla` *codant pour l'hémolysine* `alpha`*, est essentielle pour induire la traduction de cet ARNm. (d'après [KEER02, BKR+00])*

L'avantage d'un tel régulateur multi-domaines est de pouvoir coordonner la régulation de plusieurs étapes clés associées à une fonction biologique donnée. En effet, la quantité de production d'un régulateur donné influence son efficacité et si deux ou plusieurs fonctions régulatrices sont portées par une même molécule d'ARN, l'efficacité de ces dernières se trouvera ainsi équilibrée.

1.4 L'ARNomique, propriétés utilisées pour la localisation des ARN

Les premières mises en évidence de petits ARN non codants ont été basées sur des études expérimentales. Cependant, avec la croissance exponentielle des bases de données, l'approche *in vivo* devient de plus en plus laborieuse et le besoin de logiciels pour la détection de régions fonctionnelles est de plus en plus flagrant. L'approche *in silico* permet ainsi de prédire des gènes non-codants et de fournir des candidats aux biologistes. La démarche de ces derniers consiste ensuite à en démontrer expérimentalement leur synthèse et à élucider leur fonction biologique. Un nouveau champ de recherche nommé ARNomique a vu le jour ces dernières années. Il désigne les études portant sur l'identification et l'étude fonctionnelle des gènes dits non-codants. Les récentes recherches systématiques pour de tels ARN ont changé notre vision au sujet de leur prévalence et de nombreux représentants de ces molécules sont maintenant connus aussi bien chez les archae-bactéries que chez les procaryotes et eucaryotes (pour revue [Sto02]).

Alors que les outils de prédiction des gènes codants pour des protéines sont de plus en plus performants, il reste toujours très difficile de détecter les gènes d'ARN non codants. La raison principale est liée au fait que les signaux utilisés par les logiciels de recherche de gènes, tels les codons start et stop (délimitant le cadre de lecture ouvert) ou le biais spécifique de chaque espèce dans l'usage des codons, ne sont pas présents dans les gènes d'ARN. Ceux présents dans les gènes d'ARN non codants sont à l'inverse moins bien reconnaissables (promoteurs, terminateurs) et, de ce fait, représentent des indicateurs moins pertinents de la présence de gènes.

Les signaux biologiques sont généralement caractérisés par leur séquence. Cependant, pour beaucoup de processus biologiques, le vrai signal est défini par une structure spatiale d'ordre supérieur [BKV96, Edd99]. L'exemple type concerne les molécules d'ARN. En effet, en prenant l'exemple des ARN `16S`, Woese *et al* [WGGN83] ont montré par une étude phylogénétique une meilleure conservation de la structure secondaire : elle est une caractéristique plus pertinente que la structure primaire. Les programmes de référence de recherche de similarités que FASTA [PL88] ou BLAST [AGM$^+$90] sont donc limités pour l'identification de molécules d'ARN.

Les informations contenues à la fois dans la séquence et la structure peuvent ainsi être vues comme des signaux biologiques à exploiter pour une recherche. Les paragraphes suivants abordent les approches les plus utilisées pour localiser des gènes d'ARN (pour revue [Sch02a]). Les méthodologies basées sur des approches thermodynamique ou comparative permettent de faire de la recherche *de novo* d'ARN non codants en utilisant des caractéristiques générales à l'ensemble des ARN. Les deux autres approches nécessitent de connaître des molécules d'ARN modèles ou les caractéristiques spécifiques d'une famille.

1.4.1 Approche thermodynamique

Le principe d'une approche thermodynamique découle des trois hypothèses suivantes :

- à chaque configuration d'une molécule d'ARN peut être associée une quantité d'énergie libre ;
- la configuration la plus stable est supposée donnée par celle qui minimise l'énergie libre ;

– la molécule, en se repliant adopte la configuration la plus stable.

Actuellement, l'énergie libre d'une structure secondaire est calculée en effectuant la somme des énergies libres des paires empilées de bases qui la forment (modèle du plus proche voisin). Les structures tertiaires potentielles ne sont pas prises en compte (les pseudo-noeuds, triples hélices sont exclus) dans ce calcul. L'approche thermodynamique consiste à sélectionner des séquences candidates lorsque l'énergie optimale résultante de la structure secondaire de la molécule est inférieure à un certain seuil.

La prédiction du repliement des molécules d'ARN reste un des challenges de la bioinformatique. La structure secondaire de la molécule apporte la contribution principale à l'énergie de stabilisation et aux contraintes stériques qui guident le repliement tridimensionnel. Aussi, les algorithmes de prédiction de structures secondaires utilisent comme critère le minimum d'énergie libre [ZS84] pour calculer une prédiction, intègrent des informations sur le repliement cinétique [Mar84, FFHS00] et comptabilisent le nombre d'appariements [Hig96].

Il existe essentiellement deux grandes classes de méthodes de prédiction de structures secondaires d'ARN. Dans la première, on suppose que la structure secondaire correspondant à une séquence d'ARN est la plus stable (de plus faible énergie libre) des structures envisageables. Les méthodes de la seconde classe cherchent à identifier la structure secondaire commune à un ensemble de séquences d'ARN. La majorité de ces méthodes identifient les appariements conservés (covariations) d'une séquence à l'autre, ce qui nécessite la comparaison préalable des séquences.

Hofacker *et al* [HFF+98] ont combiné une approche basée sur la prédiction de structure secondaire avec une approche de comparaison phylogénétique pour localiser des ARN dans les génomes de virus.

1.4.2 Détermination du contenu en (G+C)

Les variations statistiques de la composition de bases d'un génome, tel que le (G+C)%, a été un critère largement utilisé pour rechercher des gènes codant pour des protéines (pour revue [KCM98]), mais aussi pour sélectionner les régions riches en ARN non codants [KME02, Sch02b].

Par exemple, les génomes hyperthermophiles doivent protéger leurs molécules double brin d'ADN et simple brin d'ARN contre la dénaturation thermique. La stratégie la plus simple consiste à augmenter leur (G+C)%. Cependant, le (G+C)% de génomes hyperthermophiles n'est pas corrélé aux températures optimales de croissance [GL97, Gro98, DC00]. Les génomes hyperthermophiles utilisent des mécanismes variés pour stabiliser leurs molécules d'ADN, en augmentant par exemple les concentrations ioniques intracellulaire, le super-enroulement de l'ADN [Gro98, DC00]... Les structures secondaires d'ARN intra-moléculaires semblent cependant être partiellement stabilisées par une augmentation du nombre de liaisons hydrogènes. En effet, le (G+C)% des ARN ribosomiques et ARN de transfert de génomes hyperthermophiles montrent une forte corrélation avec les températures optimales de croissance [GL97].

E. Rivas et S. Eddy [RE00b] ont proposé un algorithme de recherche d'ARNnc basé sur le (G+C)%. Ils suggèrent également d'appliquer cette méthode à des génomes non thermophiles présentant un (G+C)% variable entre les ARN non codants et le génome. Cependant, la faisabilité d'une telle démarche n'est pas encore claire.

P. Schattner [Sch02b] a pratiqué une analyse sur la variation statistique de la composition de bases comme le (G+C)%, (G-C)%, (A-T)% et les fréquences de dinucléotides sur différentes classes d'ARNnc et de trois génomes. A partir de ses conclusions, il propose une méthode de détection d'ARNnc sur la base des variations de (G+C)% et la fréquence des dinucléotides "GC" . Les résultats obtenus sur *M. jannaschii* montrent une augmentation de la spécificité de l'algorithme grâce à la prise en compte de la fréquence des dinucléotides "GC" [Sch02b].

1.4.3 Approche par modélisation comparative

Ce type d'approche regroupe l'ensemble des méthodes qui modélisent l'information portée par un groupe de séquences homologues et l'utilisent ensuite pour rechercher de nouvelles séquences. Des analyses bioinformatiques basées sur la conservation des séquences parmi des organismes proches et/ou la conservation structurelle des ARN ont montré leur efficacité pour l'identification de gènes. D'autres approches récentes chez *E. coli* exploitent les réseaux de neurones pour extraire les caractéristiques communes parmi les ARN connus [CDH01].

Les analyses intégrant ce type d'approche nécessitent essentiellement les deux étapes suivantes :

Étape d'apprentissage du modèle : recherche des signatures caractéristiques d'un ensemble de molécules d'ARN regroupées sur le critère d'appartenance à une même famille (même fonction, par exemple).

Étape de recherche de séquences satisfaisant le modèle.

Il existe essentiellement deux types d'approches pour apprendre le modèle : approche supervisée et non supervisée. Les méthodes utilisant une approche supervisée identifient les appariements conservés (covariations) d'une séquence à l'autre, ce qui nécessite l'alignement préalable des séquences, opération souvent effectuée à la main par des experts.

Approche non supervisée

A l'inverse des techniques utilisant les covariations, certaines approches (non supervisées) n'imposent aucune contrainte sur la localisation des structures au sein des séquences et permettent ainsi de s'affranchir de l'opération d'alignement. Par exemple, Bouthinon *et al* ont proposé [BD99] une méthode d' apprentissage d'une description commune (la structure secondaire) à un ensemble d'objets (les séquences) possédants chacun plusieurs descriptions possibles (les structures envisageables), une seule correspondant à la réalité. Leur représentation des structures secondaires leur permet d'envisager toutes les formes d'interactions tertiaires (pseudo-noeuds).

Approche supervisée

La comparaison (l'alignement est une réponse à ce problème, chapitre 2, page 45) permet de révéler les relations qui existent entre différentes séquences, qu'il s'agisse d'informations de structure primaire et/ou secondaire. Selon le niveau de structure pris en compte, il existe différentes façons de modéliser l'information commune à l'ensemble de ces séquences.

1.4.3.1 Modélisation de la structure primaire

La première façon de représenter les informations contenues dans un alignement est de construire un consensus où pour chaque colonne, on garde le nucléotide le plus représenté.

Une deuxième manière de faire est de construire une expression régulière, laquelle autorise une certaine ambiguïté. On perd moins d'information qu'avec le consensus. En effet, dans ce cas l'information sur la variabilité des nucléotides à chaque position est conservée. Par contre, on ne dispose toujours pas d'information précise sur leur fréquence. L'expression régulière n'est pas suffisamment discriminante.

Des méthodes probabilistes ont été mises au point pour éviter ces pertes d'information et elles prennent en compte la spécificité de la position : les matrices de fréquence pour les acides nucléiques (fréquence de chaque nucléotide trouvé à chaque position de l'alignement multiple) ou de nouvelles méthodes "profils" basées sur les modèles de Markov cachés. Ce dernier type de modélisation présente un niveau d'information supérieur au précédent et est aussi largement utilisé par les logiciels de prédiction de gènes codants.

Le chapitre II (section 2.1, page 44) décrit les caractéristiques de ces différents niveaux de représentation.

1.4.3.2 Modélisation de la structure secondaire

L'analyse comparative de séquences est une approche très efficace pour la détermination des structures secondaires et de certaines interactions tertiaires des molécules d'ARN. Les changements de bases compensatoires (Fig. 1.22) observés entre plusieurs séquences sont à la base de la prédiction des structures secondaires par les méthodes phylogénétiques. Ce phénomène s'observe au cours de l'évolution des molécules : quand une base impliquée dans un appariement mute, la base complémentaire a tendance à également muter pour préserver l'appariement et donc la structure secondaire.

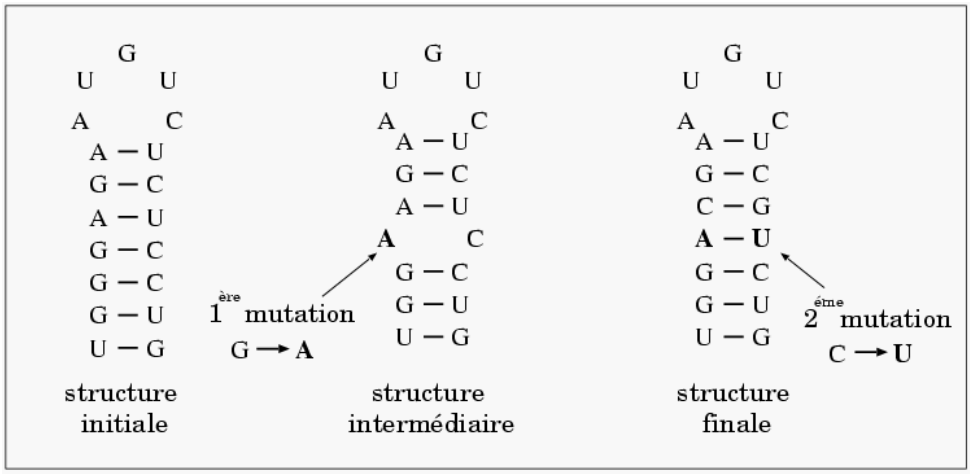


FIG. 1.22 – *Exemple de covariation ou changement de bases compensatoires au sein d'une hélice.*

Il devient donc naturel d'intégrer la structure secondaire en plus de la structure primaire aux approches de la localisation d'ARN non codants. En informatique, les grammaires (décrites

dans le chapitre 4, section 4.1, page 132) permettent de dériver les caractéristiques communes à toutes les séquences appartenant au langage correspondant. Nous verrons quelles en sont les applications possibles pour l'analyse des ARN.

1.4.3.3 Applications utilisant une approche comparative

Wassarman *et al.* [WRR+01] ont utilisé comme unique critère le degré de conservation des régions intergéniques entre *E. coli*, *Salmonella pneumonia* et *Klebsiella pneumonia* pour générer une liste d'ARNnc potentiels. Les résultats obtenus ont été relativement importants et ont donné lieu à des recherches expérimentales utilisant les *micro-arrays*. Leur méthode a permis de prédire 60 nouveaux ARNnc chez *E. coli* dont 18 ont été prouvés expérimentalement.

Une seconde approche menée par Argaman *et al.* [AHV+01] sur *E.coli* a consisté à combiner un ensemble de critères obtenus lors de la comparaison d'ARNnc déjà identifiés (entre autres les signaux transcriptionnels tel que les promoteurs et terminateurs) avec une recherche de séquences conservées parmi un ensemble de génomes bactériens. Ils ont ainsi obtenus 24 nouvelles prédictions dont 14 ont été prouvées expérimentalement. Cette approche est difficilement applicable à des génomes bactériens ou non bactériens pour lesquels les séquences signaux sont moins bien conservés entre espèces.

Rivas et Eddy [RKJE01, RE01] ont basé leur approche, implémentée dans QRNA, sur la recherche de régions conservées entre différentes espèces et sur l'examen en parallèle de la nature des différences obtenues lors de la comparaison. L'hypothèse consiste à considérer une région comme appartenant à un gène codant pour une protéine si les différences observées entre les séquences homologues correspondent soit (i) à des codons synonymes[12] soit (ii) à des codons codant pour des acides aminés ayant les mêmes caractéristiques physico-chimiques. A l'inverse, si la région contient un ARNnc, alors un plus grand pourcentage de différences surviennent entre les régions complémentaires préservant ainsi la structure secondaire des ARN (le principe des covariations). En dernier lieu, si la région ne contient aucun gène, alors la distribution des différences entre espèces devrait correspondre à leur fréquence en bases. L'avantage d'une telle approche est d'être applicable à tous génomes pour lesquels la séquence complète est disponible.

1.4.4 Approche descriptive

Contrairement à la méthode précédente, l'approche descriptive n'implique pas la connaissance de gènes homologues mais de l'ensemble des caractéristiques décrivant la molécule structurée. Les différentes méthodes inclues dans ce groupe utilisent les informations décrivant la structure primaire et/ou la structure secondaire. La qualité des signatures utilisées est dépendante des familles de molécules sur lesquelles se porte l'étude et la recherche est, de ce fait, plus ou moins efficace selon la connaissance des caractéristiques de la famille.

1.4.4.1 Exemple de séquences signatures recherchées

Les signaux les plus conventionnellement recherchés pour localiser des ARN non codants sont ceux impliqués dans les étapes de transcription. Les exemples suivants décrivent deux types

12. Le code dégénéré des acides aminés permet d'obtenir 20 acides aminés différents à partir de 64 codons.

de signaux ; ceux de type promoteur correspondant à des éléments de séquence et ceux pour les sites rho-indépendants correspondant à une structure secondaire.

Promoteur

Un site promoteur permet l'initiation de la transcription en étant reconnu par des facteurs protéiques, tels que l'ARN polymérase, l'enzyme responsable de la transcription.

Chez les eucaryotes, les ARN non codants résultent de l'activité de polymérases différentes, lesquelles reconnaissent différents types de promoteurs (section 1.1.1).

La situation est plus simple chez les procaryotes où une seule polymérase est responsable de la synthèse de l'ensemble des différents ARN. Il est alors plus simple de rechercher des signatures de promoteurs, mais les signaux correspondants souvent courts et/ou dégénérés sont peu discrimants pour une recherche n'intégrant que ce type d'informations.

Terminateur rho-indépendants

Chez *E. coli*, il existe deux possibilités pour signaler la terminaison de la transcription. Le premier système, dit rho-dépendent, fait intervenir le facteur protéique rho. Ce système n'est pas suffisamment caractérisé pour faire l'objet d'un modèle de prédiction. A l'inverse, le second système dit rho-indépendant, ou intrinsèque, est largement caractérisé. Statistiquement, environ 70% des gènes codant un ARNnc utilisent un terminateur rho-indépendant [dBT90, EKW$^+$00]

Les terminateurs rho-indépendants sont composés d'une tige boucle suivie d'une succession de bases thymine. Cette structure permet le décrochage du brin d'ARN néo-synthétisé (ARN messager).

La modélisation et la recherche de tels motifs sont facilitées par l'utilisation de logiciels de recherche de motifs, qui font l'objet du chapitre 3 (Page 69).

1.5 L'hypothèse d'un monde à ARN

Il ne m'était pas possible de finir ce premier chapitre sur les ARN en général sans parler de l'engouement actuel de la communauté scientifique pour la compréhension des origines de la vie et l'importance que l'ARN a pu y jouer. Certains scientifiques proposent des hypothèses sur le fait qu'il puisse y avoir une molécule à l'origine de la vie et à partir de cette hypothèse, ils s'interrogent sur sa nature potentielle.

Initialement, l'hypothèse dominante a d'abord été de considérer les protéines comme étant les premières molécules fonctionnelles de l'histoire du vivant. En effet, la synthèse de protéines à partir d'éléments non organiques semble relativement facile. Cependant on n'a jamais réussi à mettre en évidence des propriétés d'auto réplication chez les protéines et l'information n'a pas de moyen de se transmettre d'une génération à l'autre.

La démarche suivante a été de considérer les acides nucléiques. Le principal défaut de l'ADN pour remplir les conditions de la molécule "originelle" serait de nécessiter des protéines (en particulier l'ADN polymerase) pour être fabriqué.

La découverte des propriétés catalytiques des ribozymes a permis d'émettre la possibilité d'un monde primitif dans lequel l'ARN aurait précédé l'ADN et les protéines. L'ARN présente le double avantage de pouvoir porter et transmettre une information génétique et de pouvoir aussi catalyser les premières étapes du métabolisme. Ainsi, L'ARN est le support d'une information génétique séquentielle et est doué d'activités enzymatiques non négligeables.

Ces deux critères feraient de l'ARN [Gil86] un candidat intéressant pour le titre de "molécule à l'origine de la vie" avant que des molécules plus spécialisées telles l'ADN ou les protéines prennent le relais.

Ainsi, au cours des premières étapes de l'évolution, le transfert de l'information génétique aurait pu être assuré par la réplication de l'ARN et la catalyse aurait pu être réalisée par des petits peptides non génétiquement codés et par des ribozymes [Mau03]. De plus, l'ARN étant composé d'une seule chaîne de nucléotides, il est beaucoup plus vulnérable aux attaques des radicaux libres et aux autres agressions chimiques. Cette plus grande fragilité de l'ARN est un des arguments avancés par les évolutionnistes qui soutiennent que la vie s'est d'abord développée dans un "monde à ARN", ensuite remplacé par un "monde à ADN" plus stable [Gil86].

Les premiers arguments en faveur d'un monde ARN originel étaient :

- chez les eucaryotes les protéines sont synthétisées dans le cytoplasme là où les ARN sont en abondance, alors que l'ADN se trouve dans le noyau [BC56].
- la thymine, base spécifique de l'ADN, est obtenue par transformation de l'uracile qui est spécifique de l'ARN. Les ARN sont les amorces indispensables de la synthèse des acides désoxyribonucléiques dans l'étape de la réplication du brin *reverse* de l'ADN.
- de nombreux facteurs métaboliques sont de nature ribonucléotidique.

Les découvertes plus récentes sur l'importance des ARN fonctionnels dans la vie de la cellule ont apporté de nouveaux arguments à l'hypothèse ARN :

- Les petits ARN non codants peuvent être une relique du monde ARN originel [Edd01, MAFM01]. Le rôle indispensable dans le fonctionnement des cellules a été identifié pour un certain nombre d'ARN non codants (ARN ribosomiques, ARN de transfert, ARNase P, ARN SRP...).
- les progrès récents des méthodes de sélection *in vitro* permettent l'exploration intensive de séquences et la création de nouveaux acides nucléiques artificiels (aptamères) possédant des propriétés de reconnaissance et/ou des propriétés catalytiques. Leur production en grande quantité en font des molécules intéressantes tant sur le plan thérapeutique que sur le plan fondamental. Elle permet de mimer in vitro des processus évolutifs [MAFM01].

L'intérêt pour la localisation, la compréhension des modes d'action et l'évolution biologique des molécules d'ARN ne cesse de croître. On trouve en effet actuellement dans la littérature un grand nombre de publications faisant état de la découverte de nouveaux petits ARN non codants (ARNnc) qui contrôlent diverses fonctions biologiques essentielles [Edd01, BFPP99].

Même si tous les ARN non codants ne peuvent pas être directement considérés comme des reliques d'un monde à ARN [BFPP99], leur nombre, inconnu, et la diversité de leurs fonctions peut apparaître comme un argument en faveur d'un monde originel d'ARN.

Une autre hypothèse formulée en 1999 par Forterre et Doolittle [PF99, Doo99], au vu des spécificités biochimique et génomiques des eucaryotes, des bactéries et des archae-bactéries, signale que la théorie d'un ancêtre unique des formes vivantes est sans doute erronée : la vie serait apparue plusieurs fois, dans des environnements et des conditions différentes, et ces formes de vies primaires différentes évoluant simultanément se seraient enrichies mutuellement par des transferts de gènes. Certaines auraient disparu, trois au moins seraient à l'origine du monde vivant terrestre.

1.6 En résumé

Les ARN sont des molécules impliquées dans un grand nombre de processus biologiques. Les gènes et autres motifs fonctionnels ne peuvent être trouvés avec les “prédicteurs” de gènes codants. Certains ARN peuvent être localisés par la recherche de séquences similaires dans des génomes proches. Un certain nombre de propriétés caractérisant ces molécules peuvent être utilisées pour les localiser dans les séquences génomiques : existence d’une structure, (GC) % dans les séquences (AT) riches, (GC) % et biais en di-nucléotides, etc... Un grand nombre d’ARN interagissent avec d’autres ARN en mettant en jeu des interactions de type Watson-Crick. Un grand nombre de familles d’ARN sont caractérisées par la présence de motifs invariants et leur structure secondaire.

Chapitre 2

Recherche de mots

Nous avons décrit dans le chapitre précédent les caractéristiques des ARN auxquels nous nous intéressons. La problématique à laquelle nous souhaitons répondre porte sur la recherche de ce type de molécules dans un texte génomique. Elle appartient à un domaine important de la bioinformatique. En effet, le codage des génomes sous forme d'une chaîne de grande longueur formée par des nucléotides A, C, G, T nous permet de traiter l'ensemble des problèmes qui nous intéressent grâce à des algorithmes classiques rencontrés dans l'algorithmique de textes.

Les séquences génomiques constituent la structure primaire des génomes. La structure tertiaire est la forme fonctionnelle des molécules dans l'espace à trois dimensions. Entre ces deux formes de structures, on considère souvent la structure secondaire qui est une représentation dans le plan, étape intermédiaire entre la structure primaire et la structure tertiaire. Avant d'aborder le problème de la recherche de motifs structurés que l'on peut représenter par un ensemble de mots connectés entre eux par des relations de distance et/ou des relations sur leurs contenus (et donc par un langage chapitre 4, page 131), nous abordons dans ce chapitre la recherche d'éléments de séquence.

Cette première problématique permet déjà d'envisager de différentes manières la résolution du problème, dépendante de la représentation des éléments de séquence. Cette représentation est l'étape suivant celle permettant de mettre en relief les informations caractéristiques contenues dans un ensemble de séquences. On peut distinguer les deux étapes :

1. Extraction de l'information de plusieurs séquences ;
2. Représentation de cette information sous la forme d'un ou plusieurs mots.

La représentation d'un élément de séquence est ainsi conditionnée par le niveau de complexité de ce que l'on souhaite extraire : (i) un mot définit sur un alphabet, ou (ii) un mot appartenant à un ensemble de mots modélisés par un langage.

Pour chacune de ces représentations mots/langages, on distingue les problématiques auxquelles sont associées les différentes classes de méthodes informatiques permettant d'y répondre :

- un mot u est-il un élément de v (*string matching*) ?
- un mot u apparaît-il approximativement dans v (*alignement* ou *approximate string matching*) ?
- un des mots d'un langage X est-il un facteur de v (*pattern matching*) ?

– un des mots d'un langage X apparaît-il approximativement dans v (*approximate pattern matching*)?

Un ensemble de mots ou langage permet de représenter de façon concise un motif structuré, en intégrant selon la classe considérée des notions plus ou moins complexes de structure. Nous définissons plus spécifiquement les différents niveaux de complexité de représentation permis par les langages dans le chapitre 4.

Dans ce chapitre, nous nous intéressons uniquement à la recherche de mots appartenant à un langage fini.

Ce chapitre se focalise principalement sur les problèmes de *string matching, alignement et approximate string matching*, et les méthodes informatiques de recherche de mots utilisés dans le chapitre 5 sont détaillées dans 2.2 et 2.3.

Au préalable, nous définissons brièvement le vocabulaire nécessaire (2.1.1) et les approches existantes pour extraire (2.1.2) et représenter l'information caractéristique (2.1.3) à partir de plusieurs séquences.

2.1 Les séquences nucléiques vues comme un texte

La plupart des programmes informatiques utilisés en biologie manipulent des séquences biologiques. Un préalable commun à de nombreuses problématiques est le problème basique de la comparaison de deux ou plusieurs séquences. Dans le propos qui nous intéresse, cette problématique a pour objectif de mettre en relief les caractéristiques communes d'une famille de séquences partageant la même fonction.

La notion d'alignement de deux ou plusieurs séquences s'utilise également pour la comparaison d'acides-aminés. Dans l'ensemble des exemples considérés, nous nous contenterons d'aborder le cas particulier des acides ribonucléiques (ce qui entraîne une simplification de la notion de similarité).

2.1.1 Terminologie

Nous supposons que nous travaillons sur un texte T de longueur t dans lequel nous recherchons un mot P de longueur p (que nous supposerons plus petit que t). T et P sont décrits dans un alphabet fixé $\sum = \{A, C, G, U\}$. Le k-ième caractère d'un mot P est noté p_k. Ainsi, on représente le mot par le tableau $P[1...p]$ et le texte par le tableau $T[1...t]$.

Définition 1 (Facteur, préfixe et suffixe) *:*
Les facteurs d'un texte T ou d'un mot P se définissent comme l'ensemble des chaînes de caractères consécutifs dans le texte ou le mot. Un préfixe d'un mot P est un facteur dont le premier caractère est $P[1]$. Un suffixe d'un mot P est un facteur dont le dernier caractère est $P[p]$. Un préfixe ou un suffixe d'un mot P est dit propre s'il est différent de P.

Le résultat de la recherche d'un mot effectué sur un texte s'appelle solution ou occurrence.

Définition 2 (Occurrence d'un mot dans un texte) *:*
On dira que P apparaît dans T à la position j :

– *si* $0 \leq j \leq t - p$ *et,*

– *si* $\forall i$ *tel que* $0 \leq i \leq p - 1, t_{i+j} = p_{i+1}$ *ou encore si* $T[j...j + p - 1] = P[1...p]$

Une recherche est dite exhaustive si elle permet d'obtenir l'ensemble exhaustif des occurrences correspondant aux caractéristiques souhaitées.

Exemple

Soit $\sum = \{A, C, G, U\}$ et
$T = CGAGAUAGAGAC$
le mot $P = AGA$ possède 3 occurrences dans T aux positions 3,7 et 9

2.1.2 Extraction d'information

2.1.2.1 Motivations de la méthode comparative

"*Il faut que nous puissions d'abord avoir une idée claire de leurs caractères distinctifs et de leurs propriétés communes*"
Aristote (384-322 av JC) cité par Ernst Mayr dans son *Histoire de la biologie*

"*Il faut comparer pour rattacher les similitudes de structures aux similitudes de fonctionnement.*"
François Jacob, *La logique du vivant*, p.97, en parlant des travaux d'anatomie comparée de Cuvier et Camper.

"*Alors que la fonction ne souffre aucune fantaisie, l'organe lui, conserve quelques degrés de liberté*" François Jacob, *La logique du vivant*,p.115.

Ces citations nous permettent d'introduire la motivation de la méthode comparative. D'un point de vue biologique et informatique, l'homologie se définit de la façon suivante :

Définition 3 (Homologie en biologie) *: est homologue ce qui est hérité d'une ascendance commune.*

Cette définition est liée au constat selon lequel de nouveaux gènes apparaissent par duplication et modification de gènes plus anciens. Laissant la notion d'homologie à la biologie, nous parlerons par la suite de similarité. Par exemple, deux chaînes de caractères sont similaires si elles contiennent les mêmes caractères aux mêmes positions.

A partir de ces définitions, il devient évident que, par le biais de la comparaison de séquences présentant une même fonction biochimique, on peut mettre en évidence une ou plusieurs régions présentant un certain degré de similarité. L'alignement de séquences est un des moyen permettant d'effectuer cette comparaison entre séquences. On distingue ainsi l'alignement de deux séquences, dont on présente une méthode de résolution exacte dans la section suivante, de l'alignement multiple, présenté dans la section 2.1.2.3.

2.1.2.2 Alignement de deux séquences

Soit u et v deux mots écrits sur l'alphabet $\sum = \{A, C, G, U\}$, respectivement de longueur p_1 et p_2. On appelle alignement de u, v un couple de mots u', v' sur l'alphabet $\sum \cup \{-\}$ (où $\{-\}$ représente le caractère d'espacement ou brèche, couple qui satisfait les conditions suivantes

- les deux mots u' et v' ont même longueur ($|u'| = |v'| = l$) ;
- la suppression de "$-$" dans u' et dans v' donne u et v respectivement ;
- les brèches "$-$" ne se retrouvent pas en même position dans u' et v'.

Un exemple d'alignement de $u = CGAUUAG$ et $v = GAUCGA$ est :

```
u'  =  C G A U U - A G
v'  =  - G A U C G A -
```

A partir de deux séquences, plusieurs alignements sont possibles selon l'utilisation faite des brèches. On cherche ceux qui sont les plus intéressants dans le sens précis que l'on décrit ici :

> On définit w comme étant la fonction de similarité entre les couples de lettres. La valeur de $w(u'_i, v'_i)$ est un nombre réel positif si les lettres sont considérées semblables[13] , et on prendra des valeurs négatives ou nulles quand les lettres sont différentes, où si l'un des deux caractères correspond à une brèche : "$-$".

On considère ainsi trois opérations de base selon la nature de u'_i et v'_i, chacune donnant un score élémentaire s_i=$w(u'_i, v'_j)$:

(c) substitution : $u'_i \neq$ "$-$" et $v'_i \neq$ "$-$" ($s_i = w(u'_i, v'_i)$).

(a) insertion : $u'_i \neq$ "$-$" et $v'_i =$ "$-$" ($s_i = p$);

(b) délétion : $u'_i =$ "$-$" et $v'_i \neq$ "$-$" ($s_i = p$);

Le score de l'alignement est alors donné par : $S = \sum_{i=1}^{l} s_i$.

On distingue deux types d'alignement :

- L'alignement global qui est donné par la recherche d'un alignement de score de similarité optimal des deux séquences entières.
- L'alignement local qui est donné par la recherche de deux sous-séquences dont le score d'alignement optimal est maximal.

La détermination du score optimal d'un alignement s'effectue grâce à un algorithme de programmation dynamique, algorithme de Needleman et Wunsch [NW70] pour un alignement global et algorithme de Smith et Waterman [SW81] pour un alignement local.

Pour les deux algorithmes d'alignement exact, le temps de calcul est $O(p_1 \times p_2)$. Le temps de calcul pouvant être prohibitif dans certains cas d'analyses, des heuristiques ont été développées, tels que Blast ou Fasta. Ils présentent l'avantage d'effectuer rapidement des comparaisons mais ne garantissent pas la solution optimale.

2.1.2.3 Alignement multiple

La mise en évidence de similitude entre séquences sera renforcée si plusieurs séquences issues de plusieurs espèces partagent des éléments en commun. La méthode permettant d'ali-

13. Dans le cas d'une comparaison entre acides aminés, on oeut utiliser tout un éventail de matrice de similitude où, en général, la notion de similarité est reflétée par une valeur d'autant plus grande que les deux lettres/acide-aminés sont évalués comme partageant une similitude d'un point de vue biologique.

gner globalement ces séquences conduit à la mise en évidence de nucléotides identiques ou similaires.

Il existe deux grandes classes d'algorithmes pour réaliser des alignements multiples globaux, basés soit sur une généralisation de l'algorithme de Needleman et Wunsch, soit sur des méthodes heuristiques.

Généralisation de l'algorithme de Needleman et Wunsch

On recherche l'alignement multiple qui maximise la somme des scores de chaque alignement pour chaque paire (pour n séquences, il y a $n(n-1)/2$ paires). Cependant, la complexité spatiale et temporelle de l'algorithme dérivé est proportionnelle au produit des longueurs des séquences : si les n séquences sont de longueur L, la complexité est en $O(L^n)$. Cette complexité croit de façon exponentielle avec le nombre de séquences. Elle est donc utilisable avec un petit nombre de séquences mais ne peut répondre à la plupart des besoins.

Méthodes heuristiques

C'est l'approche la plus commune et rapide, et dans la plupart des cas, donne de bons résultats. Elle est utilisée par les programmes Clustalw [HS89, THG94] et Multalin [Cor88]. Clustalw commence par aligner deux à deux les séquences et construit l'arbre des relations évolutives entre les séquences. Les noeuds entre les branches représentent les alignements deux à deux et la racine représente l'alignement complet. Une fois cet arbre construit, le programme prend les deux séquences les plus proches et commence l'alignement multiple (l'alignement des séquences les plus proches est le plus fiable). Puis il progresse vers les séquences plus distantes, et remonte ainsi l'arbre. Ce programme est rapide pour un nombre raisonnable de séquences longues et plus lent si on aligne un grand nombre de séquences courtes. La complexité de cet algorithme est de $O(L \times n^2)$.

2.1.2.4 Alignement multiple local

L'alignement multiple local permet de mettre en évidence les régions conservées entre plusieurs séquences. Parmi les solutions possibles on distingue les méthodes dites " purement algorithmiques ", reposant sur une définition formelle précise des motifs, et les méthodes dites "d'optimisation stochastique" basées sur une modélisation statistique des motifs. Des programmes[14] comme Pratt [JCH95, Jon97], Smile [MS01] illustre la première approche et Meme [GBE96] la seconde.

2.1.3 Représentation de l'information

A partir d'un alignement multiple on peut mettre en évidence des traits caractéristiques à un ensemble de séquences. On peut ensuite extraire et représenter de différentes manières les informations mises en relief. Cette section définit l'ensemble de ces représentations, inégales en termes d'expressivité (Fig 2.2). Nous les présentons selon un gradient croissant de cette

14. Pour comparaison des différentes méthodes se référer à la plate-forme d'extraction de motifs dans des ensembles de séquences génomiques : `http://idefix.univ-rennes1.fr:8080/PatternDiscovery/`

notion. Excepté le cas du mot, qui est la représentation élémentaire de lui-même et ne nécessite pas l'étape de construction de l'alignement multiple, nous présentons un exemple concret de la représentation définie, laquelle est basée sur l'alignement multiple suivant (Fig 2.1) :

```
AUUGUACCU
AUCGUUCAU
AG -GAUCAU
AAUGUUCAU
```

FIG. 2.1 – *Alignement multiple d'un ensemble de séquences/mots.*

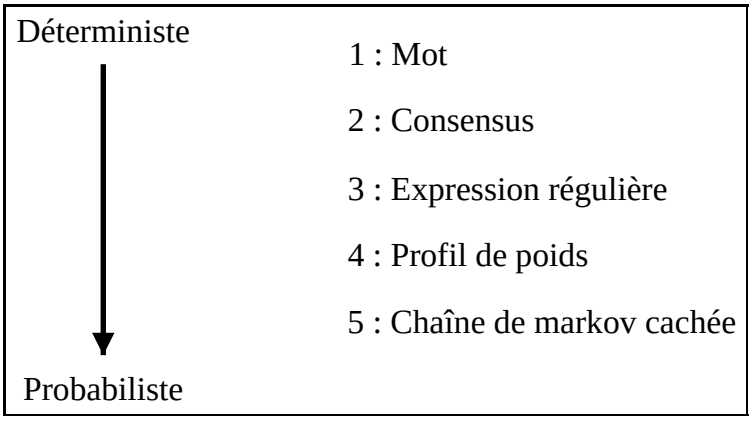


FIG. 2.2 – *Différentes modélisations d'un ou plusieurs mots selon un gradient croissant de la prise en compte de l'aspect probabiliste.*

Mot : La représentation d'un simple mot correspond à une succession de lettres.

Consensus : La séquence consensus rend compte du nucléotide le plus fréquemment rencontré pour chaque position. Dans le cas de séquences très spécifiques, cette simple séquence suffit pour décrire de manière satisfaisante une région. On peut aussi définir un consensus dégénéré sur un alphabet étendu. Par exemple, l'utilisation du code IUPAC[15] (Tab 2.1) permet de représenter une à quatre bases simultanément grâce à une liste de symboles sur l'alphabet ASCII. On appelle ainsi **classe de mots** un mot dont chaque position est une classe de caractères.

Expression régulière : Une expression régulière (Fig 2.3) permet de définir des ensembles de mots par l'utilisation d'un certain nombre d'opérations. La syntaxe pour les expressions régulières est la suivante :

- $[AG]$ autorise pour une position donnée A ou G;
- N désigne n'importe lequel des nucléotides;
- $N(4)$ désigne 4 nucléotides consécutifs;
- $N(1,4)$ indique que le nucléotide peut être trouvé 1 ou 4 fois;

15. International Union of Pure and Applied Chemistry

– – désigne une brèche ou *gap*;

A[UGA][UC-]G[UA][UA]C[AC]U

FIG. 2.3 – *Représentation de l'alignement multiple (Fig 2.1) selon une expression régulière.*

L'inconvénient majeur des expressions régulières est de modéliser des mots/séquences ne faisant pas partie de l'alignement multiple initial (par exemple, l'expression régulière de la fig 2.1 modélise 72 mots différents, alors que l'alignement multiple n'en contient que 4).

Profil de poids : Pour exprimer l'ambiguïté et la complexité d'un motif, on peut également déduire de l'alignement des séquences une table de fréquences en comptabilisant les occurrences de chaque base à chaque position du motif. En d'autres termes, on définit à partir d'un échantillon donné, la fréquence d'apparition des bases pour chaque position du motif. Il est possible ensuite, pour augmenter la fiabilité des probabilités, de considérer des critères supplémentaires, intrinsèques aux séquences, comme la fréquence attendue des bases selon la région où se trouve le motif. On peut ainsi considérer que l'apparition d'une cytosine est plus significative que l'apparition d'une guanine dans une zone riche en guanine. La transformation de la table des fréquences en tenant compte éventuellement de critères supplémentaires donne naissance à une matrice de pondération (Fig 2.4).

	A	C	G	U	–
position 1	1	0	0	0	0
position 2	0,25	0	0,25	0,5	0
position 3	0	0,25	0	0,5	0,25
position 4	0	0	1	0	0
position 5	0,25	0	0	1	0

FIG. 2.4 – *Représentation de l'alignement multiple (Fig 2.1) selon une matrice de pondération.*

code	description	code	description
A	Adenine	W	T, U, ou A
C	Cytosine	S	C ou G
G	Guanine	B	C, T, U, ou G (sauf A)
T	Thymine	D	A, T, U, ou G (sauf C)
U	Uracile	H	A, T, U, ou C (sauf G)
R	Purine (A ou G)	V	A, C, ou G (sauf T, sauf U)
Y	Pyrimidine (C, T, ou U)	N	A, C, G, T, ou U
M	C ou A	K	T, U, ou G

TAB. 2.1 – *Code IUPAC.*

HMM ou chaîne de Markov cachée : Un HMM est un modèle stochastique (probabiliste) composé d'états et de transitions les reliant entre eux. Aux états sont associées des probabilités d'émission de symboles. Le système évolue aléatoirement d'un état à un autre (les transitions sont aussi probabilistes) en émettant des symboles qui forment des mots. Les états par lesquels le système est passé ne sont pas visibles dans les mots produits, c'est pourquoi les HMM sont qualifiés de "cachés" (par opposition aux chaînes de Markov simples). Ces modèles sont des boîtes noires ; mais dans le cas d'un HMM modélisant un alignement multiple, on peut dire que les états représentent une position de l'alignement et une distribution des nucléotides sur cette position : dans ce cas, un HMM (Fig 2.5) s'apparente pratiquement à un profil, à ceci près qu'il peut comporter des états supplémentaires qui représentent des brèches de longueur variable (insertion ou deletion), auxquelles sont aussi associées des probabilités.

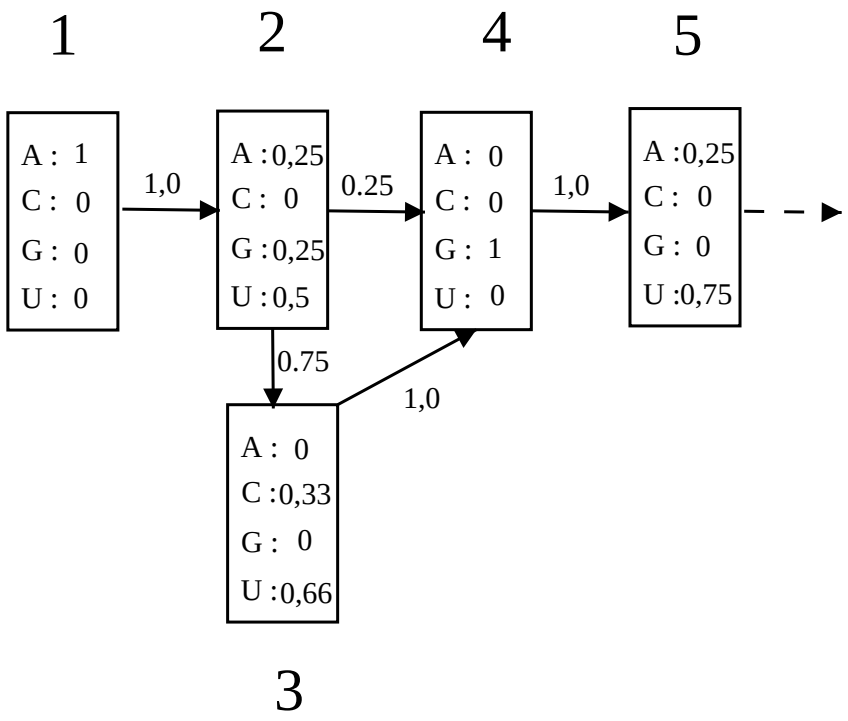


FIG. 2.5 – *Représentation des cinq premières colonnes de l'alignement multiple (Fig 2.1) selon un HMM.*

2.1.4 La recherche de mots

2.1.4.1 Introduction

L'algorithmique sur les mots est un domaine qui a suscité un intérêt considérable ces dernières années comme le montre le nombre d'ouvrages parus [CR94, CH01, Gus97]. *Algorithmique du texte* [CH01] est le premier ouvrage en français d'algorithmique spécialisé sur le traitement du texte. Il présente les bases techniques utilisées dans les domaines de la recherche documentaire, de l'indexation pour les moteurs de recherche d'informations sur l'Internet... Les

méthodes qui y sont décrites trouvent leurs applications dans les domaines du traitement de la langue naturelle, de l'analyse des séquences nucléiques et des bases de données textuelles.

Dans le cas de la recherche exacte ou approchée où l'on souhaite trouver les positions d'un mot dans un texte, on peut distinguer deux types d'approches, selon qu'on effectue ou non le traitement préalable du texte avant la recherche. La recherche approximative d'un mot est plus complexe que la recherche exacte mais répond mieux aux besoins des biologistes.

2.1.4.2 Stratégie de recherche de mots

Il existe différents algorithmes de complexité et d'efficacité variables pour résoudre ce problème.

Une première méthode consiste à effectuer un pré-traitement sur le mot à rechercher dans un texte, avant d'effectuer la recherche à proprement parler. Les deux étapes sont :

1. une phase de pré-traitement du mot où l'on obtient une fonction de décalage pour chaque lettre,
2. une phase de recherche des occurrences de P le long de T durant laquelle les données du pré-traitement permettent d'éviter des comparaisons de caractères inutiles.

L'analyse du mot recherché permet d'éviter un certain nombre de tests et ainsi, en cas de non concordance et au lieu de recommencer les comparaisons avec un décalage vers la droite, on peut effectuer un saut plus important si on sait que les positions intermédiaires ne pourront pas aboutir à une concordance. Cette approche se généralise au cas où l'on recherche non plus un seul mais un nombre donné de mots. Une autre approche complètement différente soumet le texte, plutôt que le mot, à un pré-traitement :

1. une phase de pré-traitement du texte où on obtient une structure de données,
2. une phase de recherche du mot dans la structure de données.

Cette deuxième approche repose sur des structures de données telles que les arbres de suffixes. Ceux-ci présentent comme avantage de servir d'outils d'indexation de textes, c'est à dire de fournir une représentation du texte permettant d'exécuter efficacement des requêtes diverses. La transformation d'un texte en un arbre des suffixes se fait en temps linéaire par rapport à la longueur du texte. Une fois cette représentation obtenue de nombreuses tâches peuvent être accomplies très efficacement [Gus97], par exemple la recherche d'un mot s'effectue en temps linéaire par rapport à la taille du mot. La dernière partie de ce chapitre (2.3) s'intéresse à cette approche.

2.2 Recherche avec pré-traitement du mot

L'approche classique de comparaison se base sur le parcours du texte au moyen d'une fenêtre de la longueur du mot recherché et d'un décalage de la fenêtre jusqu'à trouver ce mot. On peut explorer la fenêtre de la gauche vers la droite ou de la droite vers la gauche. La comparaison (appelée tentative) est effectuée successivement pour chaque caractère du motif tant qu'aucune concordance n'est rencontrée. Lorsque la totalité du motif a été recherchée dans la fenêtre de texte, la fenêtre de comparaison est décalée vers la droite (Fig. 2.6) après qu'un succès ait éventuellement été mémorisé.

La première classe d'algorithmes de recherche de mots exacts mémorise certaines informations obtenues pendant le déroulement de l'algorithme, afin d'éviter des comparaisons inutiles. Les algorithmes de cette catégorie diffèrent par les méthodes employées lors des tentatives de comparaisons et de décalages.

Une variante pour résoudre le problème consiste à utiliser une approche booléenne où l'efficacité est obtenue grâce à l'exploitation des capacités des ordinateurs à manipuler les mots machine, afin d'effectuer un nombre constant d'opérations à chaque position dans le texte.

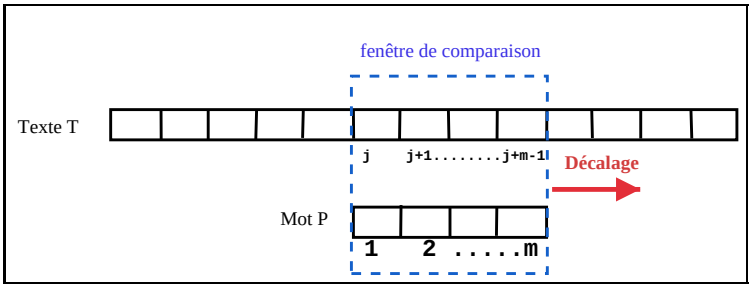


FIG. 2.6 – *Stratégie de comparaison avec une fenêtre glissante. Selon l'algorithme, la comparaison de caractère à caractère est effectuée de la gauche vers la droite ou de la droite vers la gauche.*

2.2.1 Algorithme naïf

L'approche naïve pour rechercher un mot dans un texte consiste à faire glisser la fenêtre de comparaison du mot P de gauche à droite le long du texte T. Plus formellement, pour chaque position j dans le texte, on compare $T[j..(j+p-1)]$ avec $P[1..p]$. La comparaison est arrêtée lorsque $t_{i+j} \neq p_{i+1}$, et est recommencée pour $j+1$. Une occurrence est trouvée lorsque la position $j+p-1$ est atteinte.

Lorsqu'un échec a lieu dans l'algorithme naïf, c'est-à-dire lorsqu'un caractère du motif est différent du caractère correspondant dans le texte, la recherche reprend à la position suivante en repartant au début du motif. Si le caractère qui a provoqué l'échec n'est pas au début du motif, cette recherche commence par comparer une partie du motif avec une partie du texte qui lui a déjà été comparée.

Dans l'exemple suivant, le motif $CACACC$ est trouvé dans le texte $CACACAGCACACC$ après 8 tentatives et 24 comparaisons de caractères. Les majuscules correspondent aux résultats obtenus lors de la comparaison lorsque les caractères du motif sont en vis à vis avec un caractère identique du texte, les minuscules lorsque les caractères en vis à vis sont différents (échec lors de la comparaison) et enfin les points représentent les lettres des

mots non comparés (lorsqu'un échec a été rencontré précédemment).

```
Texte :     CACACAGCACACC
Motif :     CACACc
             c.....
              CACAc.
               c....
                CAc..
                 c.....
                  c.....
                   CACACC
```

La complexité globale de la phase de recherche de cet algorithme est ($O(t \times p)$) dans le pire des cas. Cet algorithme ne nécessite aucune phase de pré-traitement du mot recherché.

Un certain nombre d'algorithmes améliorent l'efficacité de l'approche naïve en évitant d'effectuer des comparaisons inutiles. Parmi eux, certains reposent sur des méthodes effectuant la comparaison du premier caractère vers le dernier [MP70, KMP77] à l'instar de l'approche naïve et d'autres commencent la comparaison à partir du dernier caractère, comme l'algorithme décrit dans la section suivante.

2.2.2 Algorithme de Boyer et Moore

Cet algorithme [BM77] requiert une fenêtre de recherche de la longueur du mot. Cependant il explore celle-ci à partir de son dernier caractère, ce qui permet d'améliorer la vitesse de résolution. L'algorithme, au cours de la phase de pré-traitement du mot, génère deux types de tables pour y stocker les valeurs de décalages. Il utilise ensuite l'une ou l'autre pour appliquer une fonction de décalage.

La première fonction de décalage, utilisant la règle du bon suffixe, déplace la fenêtre vers la droite de manière à faire correspondre le suffixe déjà reconnu du mot dans le texte avec son occurrence la plus à droite dans $P[1..p-1]$, et si ce n'est pas possible, on essaie d'aligner le plus long suffixe du mot déjà reconnu dans le texte avec un préfixe du mot.

La deuxième fonction de décalage, utilisée parallèlement, utilise une règle définie pour chaque lettre de l'alphabet. Cette règle, appelée règle de dernière occurrence ou règle du mauvais caractère est définie pour une lettre par sa position la plus à droite dans le motif.

La complexité en temps et espace mémoire pour la génération de chaque table est de $O(p)$. L'exemple suivant illustre les deux types de décalages. Le mot $CACACC$ a été trouvé dans le texte $CACACAGCACACC$ en 3 tentatives et a nécessité 9 comparaisons de caractères.

Ces méthodes se révèlent très efficaces sur des alphabets importants. Lors de la comparai-

son, à chaque échec, on choisit la méthode qui donne le maximum de décalage.

```
Texte :     CACACAGCACACC
Motif :     .....c
```

d=2 induit par la règle du mauvais caractère

```
Texte :     CACACAGCACACC
Motif :       ....cC
```

d=5 induit par la règle du bon suffixe

```
Texte :     CACACAGCACACC
Motif :           CACACC
```

La phase de recherche a une complexité de $O(t+p)$ comparaisons. L'algorithme est d'autant plus efficace que l'alphabet est grand. Le nombre moyen de comparaisons peut alors se rapprocher dans la pratique de t/p, et est plus efficace en moyenne que l'algorithme précédent.

2.2.3 Algorithme de Baeza-Yates et Manber

Les algorithmes précédents cherchent à minimiser le nombre de comparaisons à effectuer. Une autre classe d'algorithmes se base sur une approche booléenne et présente le double avantage d'être efficaces et facile à programmer. Ces algorithmes consistent en une phase de préparation du mot et en une phase de recherche. Leur efficacité est liée à la modélisation des états de la recherche par (pour revue et comparaison de leur efficacité [EM96]) :

- un vecteur de bits (algorithme `Shift-Or` et `Shift-Add` [BYG92] et algorithme `Shift-Add` [WM91b, WM92]).;
- ou un tableau de vecteur de bits [BYP92]).

Cette représentation permet d'effectuer des opérations logiques et arithmétiques simplement. Ainsi, en adoptant une approche naïve, on obtient une recherche rapide grâce à la capacité des ordinateurs à manipuler les mots machine. Cependant, il est nécessaire que la taille du mot soit inférieure à la taille d'un mot mémoire (32 bits ou 64 bits).

Utilisant l'algorithme de Baeza-Yates et Manber [BYG92, WM91b, WM92] dans nos développements présentés dans le chapitre 5, nous le décrivons dans cette section avec plus de détails. Cette algorithme permet la recherche d'un mot en un temps d'exécution linéaire en la taille du texte. La description qui suit de leur algorithme est extraite d'un rapport technique de l'université d'Arizona [WM91b] :

P : motif à rechercher de longueur p

T : texte où effectuer la recherche de longueur t

R : tableau de bits de taille $p+1$

i : position dans le motif P et le tableau $R, i \in [1,p]$

j : position dans le texte $t, j \in [1,t]$

s_i : une lettre de l'alphabet contenue dans le motif P

S_i : un tableau de bits de taille $p+1$ pour chaque lettre s_i

S_{mask} : un tableau de bits de taille $p+1$

R_j est la valeur du tableau R après l'évaluation du caractère d'indice j du texte. Le tableau R_j mémorise la correspondance de tous les préfixes de P avec une portion du texte T se terminant à la position j. Ainsi $R_j[i] = 1$ seulement si les i premiers caractères du motif P correspondent aux caractères d'indices $j-i+1$ à j du texte t. Lorsque la lettre t_{j+1} du texte est traitée, on doit vérifier si t_{j+1} étend l'un des préfixes détectés.

La transition de l'état R_j à R_{j+1} du tableau R peut se résumer par :

$R_j[0] = 1$
pour chaque $i = 1$ *à* p **faire**
  si *(*$R_j[i-1] == 1$ *et* $p_i == t_{j+1}$*)* **alors**
    $R_{j+1}[i] = 1$
  sinon
    $R_{j+1}[i] = 0$

si *(*$R_{j+1}[p] == 1$ *)* **alors**
  alors une occurrence de P débute à la position $j-p+2$ du texte

---

Cette transition se calcule simplement sur un ordinateur : pour chaque lettre s_i de l'alphabet, nous construisons un tableau de bit S_i tel que $S_i[r] = 1$ si $p_r = s_i$ (le tableau S_i mémorise les apparitions du caractère s_i dans le motif P).

L'algorithme considère I vecteurs fixés S_i (I = nombre de lettres de l'alphabet) et le vecteur S_{mask} tels que :

$S_i[0] = 0 \ \forall \, i$
$S_i[j] = 1$ ssi s_i est la $j$ième lettre de P
$S_{mask}[0] = 1$
$S_{mask}[j] = 0 \ \forall \, j$

La transition de R_j à R_{j+1} se ramène à un décalage de un bit vers la droite de R_j et d'une opération `Et bit à bit` avec S_i (tel que $s_i = t_{j+1}$). Un exemple est donné (Fig 2.7).

Ainsi, les p opérations (lecture + comparaison) de l'algorithme sont remplacées par une lecture plus 3 opérations élémentaires sur des mots de longueur p.

Les algorithmes qui suivent une approche booléenne sont très efficaces en pratique. Lorsque le mot considéré n'est pas trop long par rapport à la taille du mot machine, la phase de recherche de l'algorithme prend un temps $O(t)$.

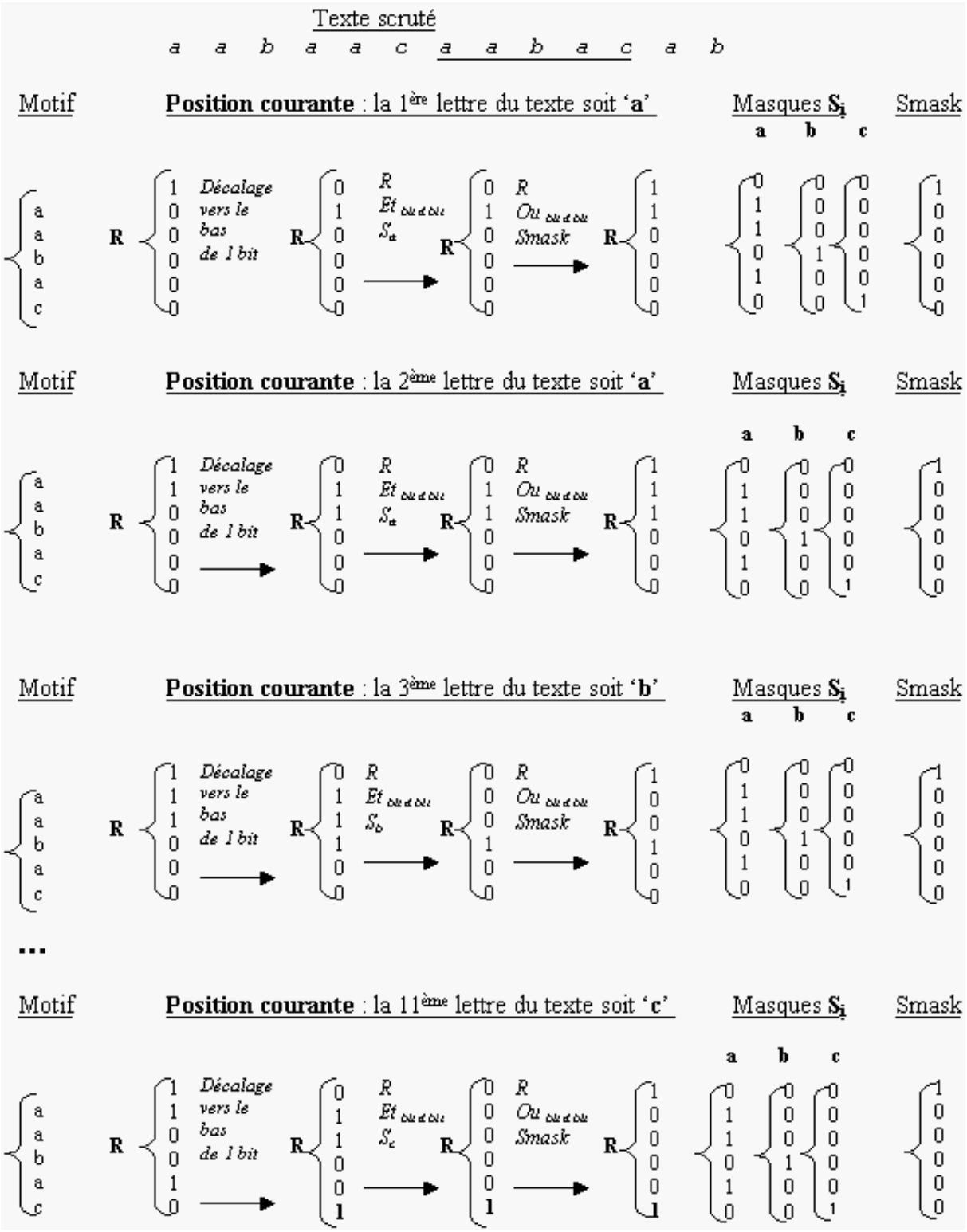


FIG. 2.7 – *Algorithme de recherche de mots sans erreur de Baeza-Yates et Manber. (d'après [Che99]). Le motif recherché est* $aabac$ *et le texte* $aabaacaabacab$*. Lorsque la 11 iéme position est considérée, un* 1 *a atteint la dernière ligne du vecteur* R*, une occurrence du motif vient d'être détectée dans le texte.*

2.2.4 Recherche approchée

La recherche approximative de mots est un problème sensiblement différent de la recherche exacte. Elle consiste à rechercher les mots en tolérant une certaine distance entre le mot et l'occurrence trouvée. On mesure généralement la distance en comptant le nombre minimum de modifications nécessaires pour passer d'un mot à l'autre : la **distance de Hamming** (Fig. 2.8). Par modification, on entend l'opération de remplacement d'une lettre par une autre. En plus de cette opération (avec une pénalité plus importante), on peut considérer l'ajout et la suppression d'une lettre, qui définissent la **distance de Levenshtein** (Fig. 2.9). Ces deux distances ne génèrent pas les mêmes résultats. Si on se réfère à un alignement entre deux mots, la **distance de Hamming** va correspondre au nombre total de lettres différentes en vis à vis. La **distance de Levenshtein** est donnée par cette première valeur à laquelle on ajoute le nombre total de brèches. Ainsi, si le calcul d'un alignement de deux mots est effectué en autorisant seulement une **distance de Hamming**, l'opération qui consiste à ajouter des brèches est interdite. Si l'opération d'ajout ou de suppression est autorisée, une pénalité (plus importante que pour la substitution) lui est adjointe et l'algorithme d'alignement cherche à minimiser son utilisation.

FIG. 2.8 – *Mots avec une distance de Hamming = 2.*

FIG. 2.9 – *Mots avec une distance de Levenshtein = 1.*

2.2.4.1 Algorithme naïf et programmation dynamique

Le problème de la recherche approchée peut être envisagé comme un problème de comparaisons ou d'alignement de séquence. Une adaptation de l'algorithme de Smith-Waterman ([SW81]) permet de prendre en considération la distance de Levenshtein avec une complexité de $O(t \times p)$ et nécessite $O(t \times p)$ en espace mémoire. Cependant, il est possible de travailler seulement avec une complexité en espace mémoire de $O(t)$. Une autre adaptation de la programmation dynamique (pour revue [NR02]), appelée DP, permet d'obtenir une complexité de recherche en moyenne de $O(k \times t)$ (avec k le nombre d'erreurs autorisées).

2.2.4.2 Approche booléenne

S. Wu et U. Manber [WM91b, WM91a] ont également conçu un algorithme de recherche approchée de mots basé sur l'approche booléenne décrite précédemment. Leur programme s'appelle $agrep$ et permet d'utiliser soit la distance de Hamming, soit celle de Levenshtein.

Distance de Hamming

Pour un mot P et un texte T de longueur m et n, on définit les tableaux R^k de valeurs binaires, où $R^k_j[i]$ vaut 1 si et seulement si les mots $P[1..i]$ et $T[j...j+i-1]$ ne diffèrent que de k substitutions au maximum.

R^0 est identique à la matrice de Baeza-Yates et Manber [BYG92, WM91b]. Si $R^k_j[m] = 1$, il existe une occurrence inexacte de P qui se termine en j dans le texte T, et qui contient au maximum k erreurs.

Pour calculer les tableaux $(R^l)_{0 \leq l \leq k}$, on calcule toutes les colonnes j, puis on effectue le même calcul pour les colonnes $j+1$. La colonne j du tableau R^k est calculée par :

$$R^l_j = \underbrace{R^{l-1}_j}_{(1)} \textbf{ OU } \underbrace{R^l_{j-1} \textbf{ ET } S_j}_{(2)} \textbf{ OU } \underbrace{R^{l-1}_{j-1}}_{(3)}$$

Ainsi, les i premiers caractères de P correspondent, avec au plus l erreurs, à une sous-chaîne de caractères de T se terminant en j si et seulement si :

(1) : les i premiers caractères de P correspondent à une sous-chaîne de caractères de T se terminant en j avec au plus $l-1$ erreurs.

(2) : ou les $i-1$ premiers caractères de P correspondent à une sous-chaîne de caractères de T se terminant en $j-1$ avec au plus l erreurs, et $p_i = t_j$.

(3) : ou les $i-1$ premiers caractères de P correspondent à une sous-chaîne de caractères de T se terminant en $j-1$ avec au plus $l-1$ erreurs.

L'algorithme est très efficace lorsque le mot P est suffisamment petit pour que le tableau R^l puisse tenir sur un petit nombre de mots machine. La complexité de l'algorithme est en $O((k+1) \times t)$. Le nombre d'erreurs autorisées étant en pratique faible, la complexité de l'algorithme est $O(t)$.

Distance de Levenshtein

De la même façon, on peut générer un algorithme prenant en compte les erreurs du type insertion/délétion. Le calcul d'une colonne est encore une opération logique sur des mots machine. La complexité est du même ordre de grandeur que pour l'algorithme précédent.

Dans le chapitre 5 nous décrivons l'intégration de l'algorithme de Baeza-Yates et Manber dans nos développements. Son utilisation nous permet de prendre en considération, par une même approche, le problème de recherche de mot exact, approché selon la distance de Hamming ou Levenshtein.

En plus de sa simplicité, l'algorithme `Shift-add` présente l'avantage d'être flexible et facilement adaptable à la recherche de classes de mots. La prise en compte de telles classes de mots se fait simplement en modifiant l'algorithme de remplissage de S_mask (2.2.3) : à une position j dans le mot P et pour tout $s_i \in \sum$, $S_i[j]$ sera égal à 1 si s_i appartient à la classe de caractères qui correspond à cette position dans P.

Exemple

Soit $P = AGMWSU$ dans le code $IUPAC$.

Sur l'alphabet $\sum = \{A, C, G, U\}$, $P = AG[AC][AU][CG]U$.
Le tableau S_i mémorisant les apparitions du caractère s_i dans le mot P s'écrit alors :

s_i	A	C	G	U
S_i	101100	001010	010010	000101

2.3 Recherche avec pré-traitement du texte

Une autre approche pour rechercher un mot dans un texte de façon relativement efficace est d'effectuer un pré-traitement non plus sur le mot à rechercher mais sur le texte. La structure de données la plus simple pour représenter l'ensemble de tous les facteurs d'un texte est un arbre de tous ces facteurs. L'utilisation de cette structure permet d'avoir des traitements performants pour les applications utilisant un index des facteurs d'un texte. Ainsi, on peut trouver si un mot appartient à un texte simplement en cherchant si ce mot est un chemin de l'arbre représentant les facteurs du texte.

2.3.1 L'arbre digital des suffixes

2.3.1.1 Définition

Tout facteur d'un mot peut être vu comme préfixe d'un suffixe de ce mot. Par exemple, si $T = UACUACA$, l'ensemble des suffixes est représenté dans le tableau Tab 2.2.

Position	Suffixe
1	UACUACA
2	ACUACA
3	CUACA
4	UACA
5	ACA
6	CA
7	A

TAB. 2.2 – *L'ensemble des suffixes de T.*

L'arbre digital des suffixes de T (trie) (Fig 2.10) est une structure de données simple qui permet de représenter l'ensemble F de tous les suffixes d'un texte et d'accéder à tous ses facteurs et à toutes leurs occurrences de façon exhaustive. Cet arbre possède des arcs qui ont pour étiquette un ou plusieurs caractères du texte. Cet arbre est dit déterministe car tous les arcs partant d'un noeud ont une étiquette différente et correspondent à un suffixe différent. Tous les facteurs d'un mot sont représentés par un chemin unique dans cet arbre allant de la racine à un noeud intermédiaire ou une feuille.

Dans le pire des cas, la taille de cet arbre est quadratique $O(n^2)$: elle correspond à la somme des longueurs de tous les suffixes

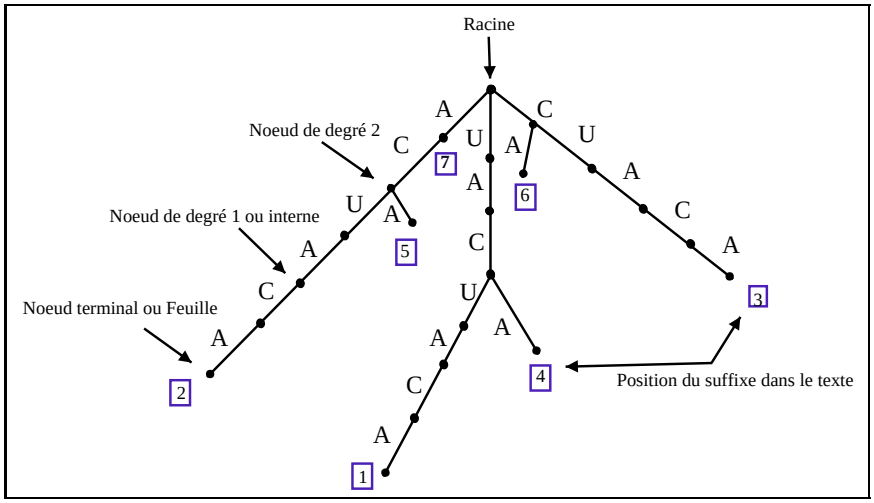


FIG. 2.10 – *L'arbre digital des suffixes pour* $UACUACA$.

2.3.1.2 Construction

Soit $T = [1...t]$ un texte donné, on note $S_i = [i...t]$ le suffixe de longueur $t - i + 1$ de T. Une façon simple de mettre le texte sous la forme d'un arbre digital des suffixes consiste à construire de façon incrémentale un arbre en insérant successivement les suffixes de longueur t à 1 (Fig 2.11). Le suffixe courant est inséré dans l'arbre à chaque étape.

Pour insérer S_i, on commence par le décomposer en $S_i = [tete_i, queue_i]$ où $tete_i$ est le plus long préfixe de S_i déjà représenté dans l'arbre. La branche $queue_i$ est alors insérée dans l'arbre, à la suite du dernier caractère du chemin $tete_i$.

La recherche du noeud correspondant à la tête est effectuée par une procédure de recherche en profondeur dans l'arbre où chaque caractère du suffixe à insérer est comparé caractère à caractère. La complexité de la recherche du noeud correspondant à l'endroit où devra être insérée la queue de S_i est $O(|tete_i|)$. Ensuite, l'ajout de la $queue_i$ entraîne la création d'un noeud et d'un arc pour chaque caractère de $queue_i$, et la complexité de cette étape est de $O(|queue_i|)$.

La complexité de l'insertion d'un suffixe est donc de $O(|S_i|)$, ce qui donne pour l'ensemble du texte une complexité de $O(|S_1| + |S_2|... + 1) = O(t^2)$.

Afin que la présence d'un suffixe étant également un facteur dans la séquence soit signifiée, on ajoute un symbole ne figurant pas dans l'alphabet ($\sum$) à la fin du texte qui sert à construire l'arbre.

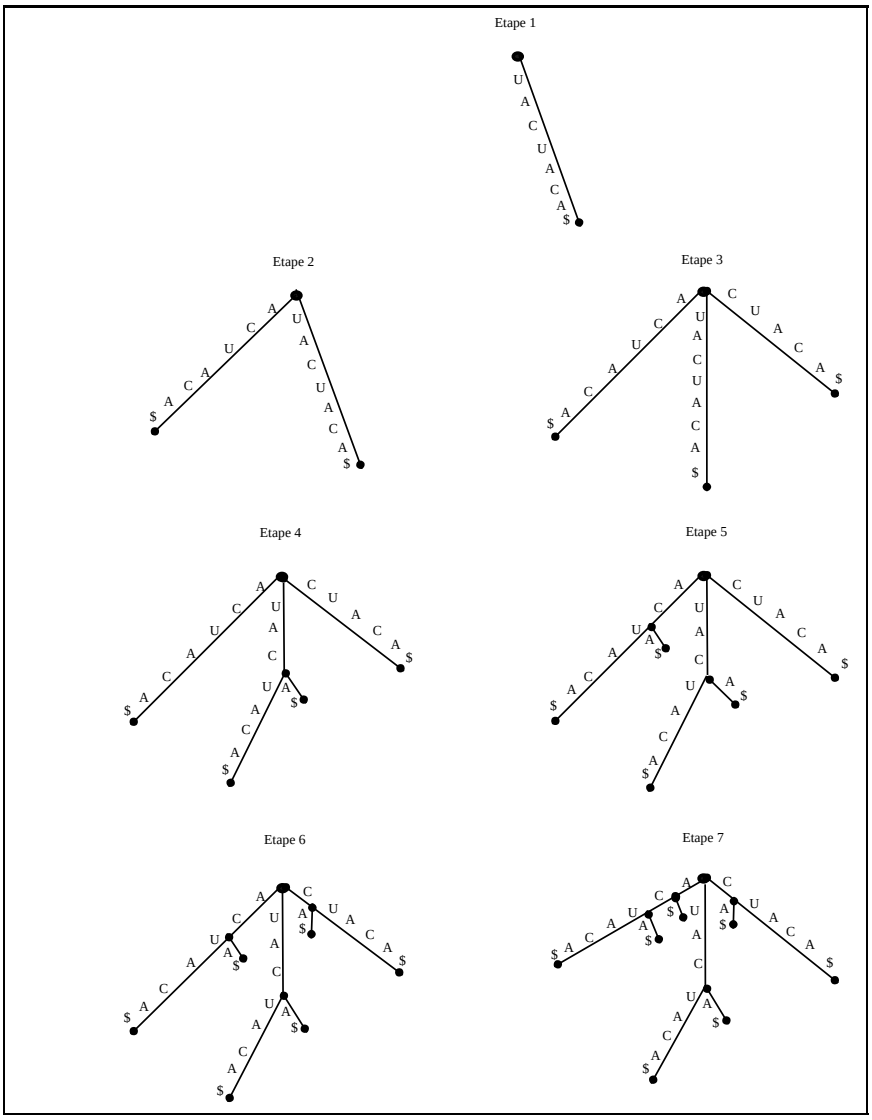


FIG. 2.11 – *Les différentes étapes de la construction de* **l'arbre digital des suffixes** *du texte* $UACUACA$.

2.3.2 L'arbre des suffixes

La représentation des suffixes d'un mot par l'arbre digital des suffixes est simple mais présente l'inconvénient majeur de nécessiter un espace mémoire quadratique par rapport à la taille du texte. Weiner a proposé une structure compacte de l'arbre digital des suffixes, appelée **arbre des suffixes**. Il a définit une structure de données pour représenter l'ensemble des suffixes d'un texte (Fig 2.12), en supprimant tous les noeuds internes de l'arbre ne donnant pas naissance à deux branches différentes ou ne correspondant pas au caractère terminal d'un suffixe. Ainsi les étiquettes des arcs peuvent être composées de plusieurs lettres. Cette compression permet d'obtenir un arbre de taille linéaire par rapport à la taille du texte. De plus, il a proposé un algorithme linéaire pour construire cet arbre (utilisant les liens suffixes), amélioré ensuite par McCreight [McC76].

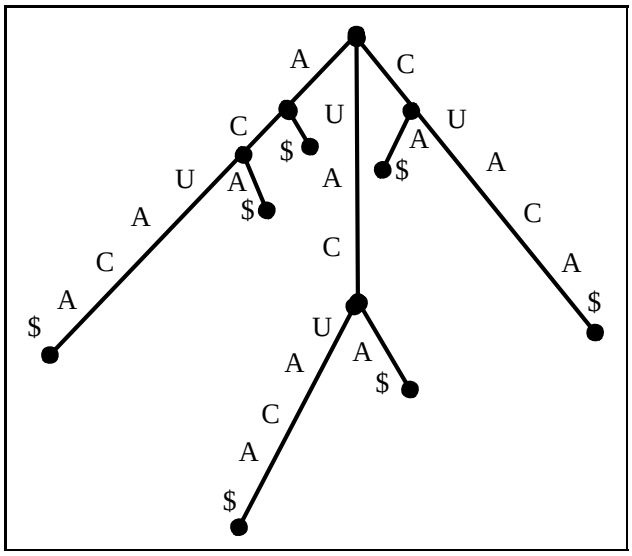


FIG. 2.12 – **Arbre des suffixes** *du texte* $UACUACA$*, version compacte de l'***arbre digital**.

Le nombre de noeuds est linéaire. Cependant, cette linéarité n'implique pas à priori celle de la représentation de l'arbre compact des suffixes car celle-ci dépend aussi de la taille des étiquettes des arcs. Néanmoins, comme ces étiquettes sont des facteurs de T, chacune peut être représentée par la valeur correspondant à sa position dans le texte et la valeur donnant sa longueur. Ainsi, il suffit de stocker au préalable une seule fois la totalité du texte en mémoire et d'utiliser ensuite ces couples de valeurs associées à chaque noeud de l'arbre pour avoir un accès performant à tous les suffixes du texte (Fig 2.13).

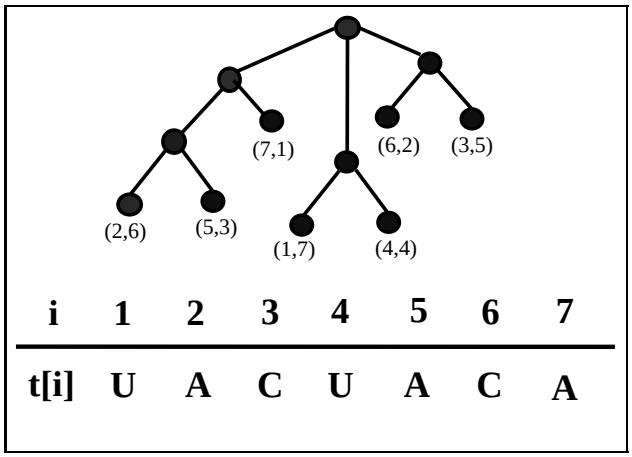


FIG. 2.13 – *Représentation des étiquettes dans l'arbre compact des suffixes du texte* $UACUACA$*. Ainsi l'étiquette* $(3,5)$ *représente le facteur de longueur* 5 *à la position* 3 *:* $CUACA$

2.3.2.1 Définition

Un arbre des suffixes (AS) du texte T est un arbre enraciné à n feuilles tel que :

- Ses noeuds internes sont de degré > 1.
- Chaque feuille représente un seul et unique suffixe $T[i...t]$ et est étiquetée par $(i, (t-i+1))$.
- Ses arcs sont étiquetés par des mots non vides.
- Les étiquettes des fils d'un noeud débutent toutes par un caractère différent.
- La concaténation des étiquettes sur le chemin de la racine r jusqu'à une feuille i forme le suffixe $T[i...t]$.
- Aucun suffixe n'est préfixe d'un autre.

2.3.2.2 Construction

Les deux algorithmes de construction de l'arbre des suffixes les plus utilisées aujourd'hui sont :

L'algorithme de McCreight [McC76]

Cet algorithme est considéré, avec celui de Weiner, comme l'algorithme de base de construction des arbres de suffixes.

La construction de l'arbre est accélérée en améliorant la recherche du noeud correspondant à l'embranchement de la tête et de la queue du suffixe en cours d'insertion. Pour cela, Mc Creight a défini la notion de lien suffixe [McC76] où pour chaque noeud de l'arbre p_1 correspondant au facteur xy, il y a un lien suffixe d'un noeud p_1 vers un noeud p_2 si ce dernier correspond au facteur y (Fig. 2.14). Ce sont les propriétés des liens suffixes qui permettent d'obtenir la linéarité en temps pour la construction de l'arbre.

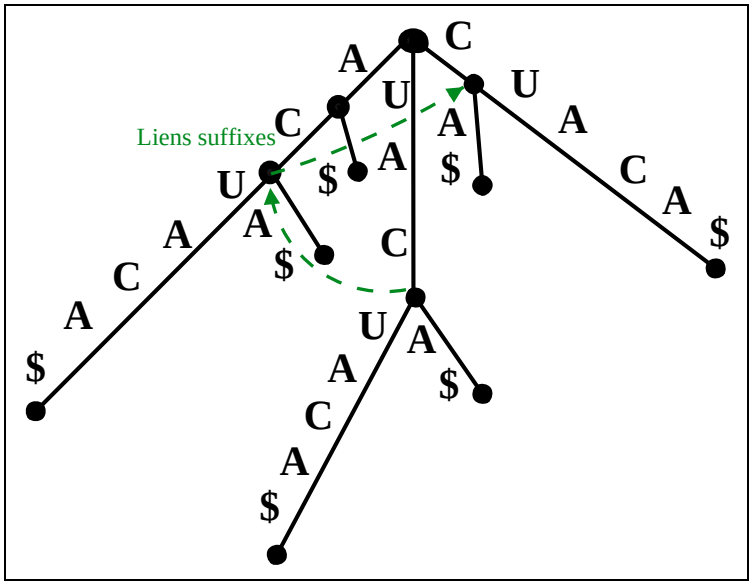


FIG. 2.14 – *L'ensemble des liens suffixes de* **l'arbre digital des suffixes** *du texte* $UACUACA$

L'algorithme d'Ukkonen[Ukk92]

Cet algorithme repose sur l'idée d'une construction incrémentale de l'arbre des suffixes implicite (un arbre des suffixes implicite est un arbre des suffixes sans la contrainte que chaque suffixe soit associé à une feuille). L'algorithme construit une suite $I_0, I_1, ..., I_n$ d'arbres de suffixes implicites pour les préfixes du texte $T[1,1], ..., T[1,n]$. Cette étape se décompose en n phases pour i allant de 1 à n. La phase i insère ensuite le préfixe $T[1,i]$ dans l'arbre des suffixes implicite I_{i-1} pour créer I_i. Au cours de chaque phase i sont effectuées I extensions pour les suffixes $T[1,i], ..., T[I,i]$ de $T[1,i]$. L'extension j insère le suffixe $T[j,i]$ du préfixe courant de l'arbre des suffixes implicite.

2.3.2.3 Un arbre plus léger

Pour un certain nombre d'applications utilisant les arbres de suffixes, la longueur des mots à rechercher dans l'arbre des suffixes peut être limitée. Des exemples peuvent être donnés par les méthodologies d'extraction de motifs de séquences biologiques [MS01, AS03] ou encore pour une application de motifs structurés, à laquelle nous nous intéressons. En effet, si la taille du mot recherché dans l'AS est comprise entre k_{min} et k_{max}, il n'est plus nécessaire d'avoir l'intégralité de l'arbre des suffixes. Un arbre de tous les facteurs dont la longueur est comprises entre ces deux bornes suffit (Fig 2.16).

Nous avons ainsi utilisé dans le cadre de nos développements exposés dans le chapitre 5 (page 149) la structure de données proposées par J. Allali et MF. Sagot [AS03]. Cette structure de données, appelée *k-factor tree* par les auteurs, possède les propriétés linéaires des arbres de suffixes. La construction de cet arbre suit l'implémentation proposée par Ukkonen [Ukk95]. L'algorithme [AS03] permet d'obtenir un gain d'efficacité au niveau de la construction de l'arbre, mais aussi lors de son parcours pour la recherche de mots.

Ainsi, si l'on considère le texte $UACUACA$ et que l'on souhaite y rechercher les mots d'une taille donnée *a priori*, par exemple 4 lettres, on obtient alors les facteurs de longueur maximale 4 du tableau 2.3, représentés dans l'arbre des facteurs de la figure 2.15 (à comparer avec la figure 2.12).

Position	Suffixe
1	UACU
2	ACUA
3	CUAC
4	UACA

TAB. 2.3 – *L'ensemble des facteurs de longueur maximale 4 du texte $UACUACA$.*

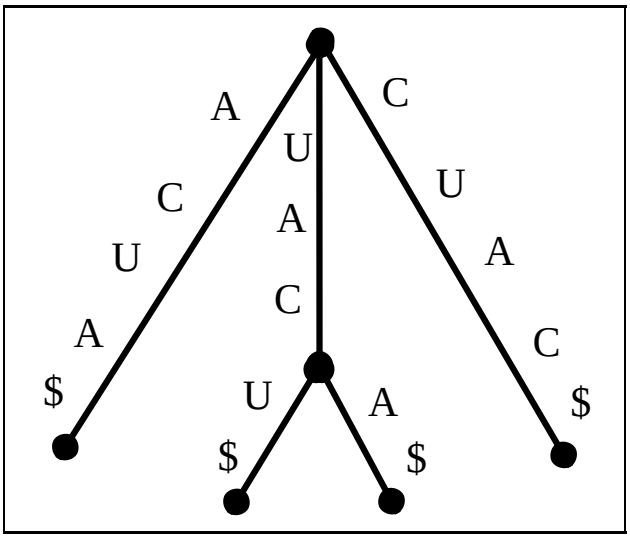


FIG. 2.15 – *L'arbre des facteurs de longueur maximale 4 du texte* $UACUACA$.

2.3.3 Recherche d'un mot dans un arbre des suffixes

L'une des applications utilisant les arbres des suffixes est de permettre de façon efficace la vérification de la présence d'un mot. Les sections suivantes décrivent brièvement les principes d'une recherche exacte ou approchée.

2.3.3.1 Recherche exacte

A partir d'un arbre des suffixes, la recherche de l'ensemble des occurrences d'un mot consiste à rechercher, s'il existe, le chemin dans l'arbre correspondant au mot. La position initiale de chaque suffixe peut être stockée au niveau de la feuille ou du noeud correspondant à ce dernier. Ainsi, s'il n'existe pas dans le texte, et correspond à un noeud interne, ou à une feuille, ou encore au préfixe d'un arc, l'ensemble de ses occurences est donné par l'ensemble de ses feuilles filles où les positions initiales sont mémorisées. La complexité de la recherche d'un mot est de $O(p \times log \mid \sum \mid)$ (dans le cas de séquences nucléiques, $\mid \sum \mid = 4$).

Dans l'exemple donné (Fig 2.16), la recherche du mot UAC dans l'AS aboutit au noeud interne entouré d'un cercle. Les deux occurrences correspondantes sont données par les deux feuilles filles du noeud.

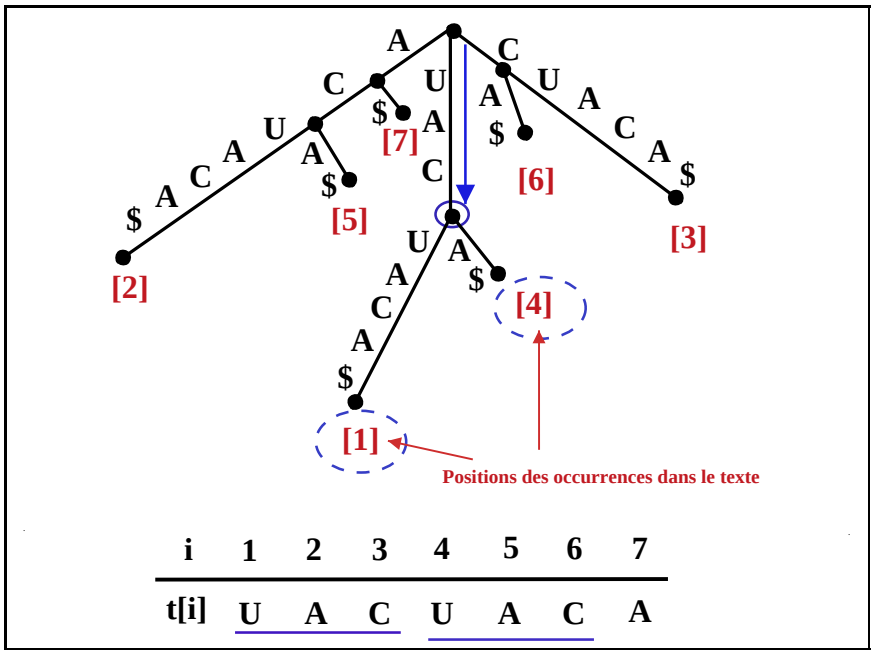


FIG. 2.16 – *Le texte $UACUACA$ contient deux occurrences du mot UAC aux positions 1 et 4.*

2.3.3.2 Recherche approchée

Il est également possible de rechercher de manière approchée un mot dans un arbre de suffixes. Le principe consiste à trouver les indices de début i de tous les facteurs de T tels que la distance entre les facteurs et le mot recherché soit inférieure ou égale au nombre d'erreurs autorisées. La recherche est effectuée en explorant en profondeur l'arbre des suffixes à partir de la racine et a une complexité exponentielle dans le nombre d'erreurs.

2.3.3.3 Applications

L'arbre des suffixes présente l'avantage primordial d'avoir une taille linéaire par rapport au texte de base. De plus, la majorité des traitements envisageables sur le texte s'exécutent aussi en temps linéaire. Enfin, le temps d'accès à un facteur du texte se fait en temps linéaire par rapport à la taille de ce facteur.

Par la suite, à cause de sa linéarité en espace et en temps de calcul, cette structure a suscité un vif intérêt et a donné lieu à de nombreuses améliorations. Par exemple, Gusfield [Gus97] cite plusieurs améliorations possibles pour des applications spécifiques :

- Recherche de plusieurs mots;
- Recherche de mots répétés;
- Recherche du mot complémentaire, complémentaire inversé;
- Recherche de la plus longue sous-chaîne de caractères commune à deux séquences, voire x séquences;
- Statistique de texte;
- Identification des répétitions maximales.

2.3.4 Tableau de suffixes

La mémoire occupée par l'AS est un paramètre important influant sur les avantages de telles structures de données. Bien que la taille et la profondeur d'un AS soit linéaire en la taille du texte, la constante de la complexité peut être élevée. Une des premières implémentations de l'AS [McC76] nécessite ainsi 28 octets par caractère. De nombreux efforts ont été effectués durant ces dernières années pour réduire cet espace. En particulier, Kurtz [Kur99] a proposé des implémentations nécessitant 20 octets par caractère dans le pire des cas et 10 en moyenne.

Le tableau des suffixes (*suffix array*) proposé par Manber et Myers [MM93a] est une structure alternative de l'AS classique permettant de réaliser un gain de mémoire très intéressant. Basée sur un concept différent des AS, cette structure a l'avantage d'être 3 à 5 fois fois plus compacte que les AS. Pour comparaison, le tableau de suffixes peut être construit en temps $O(t \times log\ t)$ dans le pire des cas et $O(t \times log\ log\ t)$ en moyenne.

Cette structure est composée d'un vecteur $Position$ des positions de tous les suffixes du texte triées dans l'ordre lexicographique. Le tableau de la table 2.4 est à comparer avec l'ensemble des suffixes de la table 2.2. $Position[k]$ contient ainsi la position de départ du $k$ième plus petit suffixe du texte dans l'ordre lexicographique. A partir du vecteur de positions, la recherche d'un mot se fait par recherche dichotomique avec une complexité de $O(p \times log\ t)$ dans le pire des cas. Cependant on peut accélérer la recherche (complexité $O(p + log\ t)$ par l'utilisation d'un second tableau $LCPs$. On obtient ainsi une recherche en moyenne quasiment aussi rapide qu'en utilisant un AS avec une structure beaucoup plus petite.

Le domaine est en plein essor actuellement, et en 2003, trois équipes, inspirés par l'algorithme compliqué de Farach [M97], ont publiés des travaux permettant de construire en temps linéaire les tableaux de suffixes [KS03, KA03, KSPP03]. La recherche d'un mot peut être également effectuée en temps linéaire [AOK02, SKPP03].

Position lexicographique	Suffixe
1	A$
2	ACA$
3	ACUACA$
4	CA$
5	CUACA$
6	UACA$
7	UACUACA$

TAB. 2.4 – *Tableau des suffixes du texte* $T = UACUACA$.

2.4 Conclusion

Le type d'approche pré-traitement du mot ou du texte avant d'entamer la recherche) pour effectuer la recherche d'un mot dans un texte peut être choisi selon le contexte de la recherche.

En effet, si l'on souhaite rechercher les occurrences d'un mot connu *a priori*, il est plus judicieux d'effectuer un pré-traitement initial et unique sur ce dernier avant de le rechercher dans le texte.

Par contre, (i) si ce mot n'est pas connu *a priori*, mais déterminé dynamiquement au cours de la recherche, et (ii) si aucune information sur sa localisation n'est connue, l'utilisation d'une structure de donnée pour représenter le texte est intéressante pour optimiser l'efficacité de la recherche de mots. La construction d'un arbre de suffixe à partir du texte est alors effectuée à l'initialisation de la recherche une fois pour toute. Si la taille des mots est connue *a priori*, on peut alors gagner en efficacité par rapport à la construction, au stockage de l'AS et sur la recherche des mots en utilisant un arbre plus léger [AS03].

Dans le chapitre 5 (page 149), les deux contextes de recherche brièvement décrits sont envisagés et ont induit l'utilisation parallèle de plusieurs approches de recherche de mots. De manière plus précise :

- dans le cas de la recherche exacte ou approchée d'un mot connu *a priori*, nous utilisons l'algorithme de Manber et Myers.
- dans le cas de la recherche de structure secondaire, mettant en jeu une relation sur le contenu de deux mots, nous considérons deux cas de figure. La recherche d'une structure secondaire consiste à partir d'une première région donnée R_1 à rechercher un mot dans une seconde région R_2 dont le contenu doit être dépendent d'une sous région de R_1. Nous distinguons deux niveaux de difficultés :
  - si la recherche de structure secondaire est effectuée au sein d'une même molécule, la région R_2, où chaque mot (sur lequel un traitement peut être fait) d'une taille donnée de R_1 est recherché, est bornée par R_1. L'approche utilisée consistera alors à inspecter naïvement chaque caractère de R_1 et de chercher dans un R_2 le caractère qui peut lui être associé. structure secondaire est au sein d'une même molécule, la complexité de la recherche est diminuée en pratique.
  - si la recherche de structure secondaire met en relation deux molécules, la région R_2 n'est plus bornée par R_1. Ce cas de figure (recherche d'interactions inter-moléculaires) est plus coûteux en terme de recherche. Nous utilisons une structure de donnée de type AS pour représenter l'ensemble des facteurs de taille k de R_2 .

Chapitre 3

Étude de l'existant dans le cadre CSP

Nous présentons dans ce chapitre une étude dont l'objectif est d'appréhender les particularités et les possibilités des logiciels utilisés dans le cadre de la recherche de motifs d'ARN. La sélection d'outils que nous avons réalisée n'est pas exhaustive. Elle reflète principalement le degré d'utilisation et la disponibilité des différents outils dans le cadre précis de la recherche de motifs d'ARN. Les outils de recherche de motifs structurés s'organisent en essentiellement deux grandes classes : les logiciels spécifiques et les logiciels généralistes.

Les logiciels dits **famille-spécifiques** sont dédiés à une famille particulière. Un outil de ce type est spécifique à une famille de molécules (ARNt, snoARN,...), il intègre dans son code aussi bien la description du motif recherché que l'algorithme qui réalise la recherche. La deuxième classe regroupe des outils dits **généralistes** qui offrent d'une part un langage approprié permettant à un utilisateur de décrire le motif, d'autre part le code de l'algorithme qui réalise la recherche. La plupart du temps les outils existants associent le problème de la recherche de motifs à la thématique des langages et grammaires formels. En pratique cependant, le problème générique de la recherche de motifs peut être traité comme un problème de satisfaction de contraintes. C'est donc dans le cadre formel des problèmes de satisfaction de contraintes (CSP) que nous introduirons le problème. Une première tentative de formalisation a été réalisée par Eidhammer *et al.*, d'une part dans le cadre de la programmation logique par contraintes, d'autre part dans le cadre CSP [EGJR01].

La première section de ce chapitre est un préambule à cette étude. Nous y proposons une définition de ce que nous appelons un motif structuré et nous introduisons les deux classes de logiciels utilisés.

La seconde section présente essentiellement les logiciels généralistes en les "plongeant" dans le cadre formel des problèmes de satisfaction de contraintes (CSP) que nous introduirons. Nous décrirons pour chacun, lorsque ce sera possible, les variables et les contraintes qu'il est possible de définir via le langage utilisateur proposé. Nous tenterons aussi de mettre en évidence quelques principes (efficaces ou non) des algorithmes de recherche utilisés.

La troisième section de ce chapitre présente une analyse expérimentale des différents logiciels de recherche de motifs. Cette étude a porté sur deux familles de molécules d'ARN, les ARN de transfert et les petits ARN nucléolaires à boîtes C/D.

Les ARN de transfert constituent une famille de gènes caractérisée aussi bien sur le plan de sa séquence que de sa structure. Cette famille largement étudiée nous a semblé représenter un motif structuré maintenant classique et recherché systématiquement dans les phases d'anno-

tation. Les ARN (Fig. 3.2) de transfert possèdent les caractéristiques suivantes :

- Une petite taille : entre 75 et 96 nucléotides ;
- Quelques bases semi-invariantes ou invariantes (bras accepteur identique : CCA) ;
- Structure secondaire en feuille de trèfle ;
- Tiges et boucle de tailles relativement constantes.

Les petits ARN nucléolaires à boîtes C/D (Fig .3.5) constituent la deuxième famille à laquelle nous nous sommes intéressés, d'une part parce qu'elle a déjà suscité le développement d'un logiciel snoARN-spécifique, d'autre part parce que cette famille présente certaines caractéristiques impossibles à spécifier à partir des outils généralistes existant. Les caractéristiques de cette famille sont :

- la présence de motifs en séquence dits boîtes C (RUGAUGA) en début de séquence et D (CUGA) en fin de séquence ;
- la présence d'une hélice terminale optionnelle ;
- la présence de motifs optionnels en séquence dits boîtes C' (UGAUGA) et D' (CUGA) à l'intérieur de la séquence ;
- la présence d'une région susceptible de s'apparier avec une autre séquence d'ARN en amont de la boîte D (D').

Une première partie de notre travail a porté sur la recherche d'ARN de transfert, en menant parallèlement deux types d'analyses. Nous nous sommes d'abord intéressés aux logiciels généralistes en comparant leur comportement à partir d'un descripteur de molécule "équivalent". Une seconde analyse a ensuite été menée sur les logiciels spécifiques avec l'objectif de les comparer en terme de spécificité et de sensibilité.

Dans un second temps, nous nous sommes intéressés à la recherche de petits ARN nucléolaires en adoptant la démarche d'un biologiste. Les recherches de ces ARN ont été réalisées en utilisant et comparant l'utilisation de deux logiciels généralistes et d'un logiciel spécifique avec l'idée sous jacente de tendre vers les meilleurs sensibilité et spécificité.

Enfin, nous présentons une analyse plus fouillée des résultats obtenus dans le cadre de la recherche des snoARN dans la séquence génomique de *Pyrobaculum aerophilum*. Cette analyse nous a permis de proposer environ 40 candidats supplémentaires différents des 51 disponibles au NCBI [16].

Nous concluons ce chapitre par une discussion mettant en évidence une liste d'objectifs sur lesquels repose le sujet de la thèse. Ils seront présentés de façon plus détaillée dans le chapitre suivant.

3.1 Définitions

3.1.1 Motifs structurés

Nous appelons motif structuré un motif satisfaisant les caractéristiques suivantes :

- il est composé d'unités élémentaires qui sont des éléments de séquence et/ou des éléments de structure. Dans le cas des ARN, les élément de structures sont donnés par la description

16. National Center for Biotechnology Information. URL : http://www.ncbi.nlm.nih.gov/

des interactions existant à la fois au niveau de la structure secondaire (ces élément seront appelés hélices par la suite) et tertiaire (ces élément seront appelés pseudo-noeuds par la suite).

– ces unités élémentaires sont contraintes par un ensemble de caractéristiques (contenu, taille, composition, erreurs tolérées) et sont reliées entre elles par des contraintes spécifiant leur organisation dans le motif (ordre, distance,...) .

Un exemple de motif structuré est donné par la figure 3.1. Ce motif est composé d'une hélice dont la taille peut varier entre 3 et 4 paires de bases, et de trois éléments de séquence correspondants à (i) la boucle de l'hélice dont la taille peut varier entre 4 et 6 bases, (ii) un segment, intervalle de taille fixe entre l'hélice et le tri-nucléotide ACA et (iii) le tri-nucléotide.

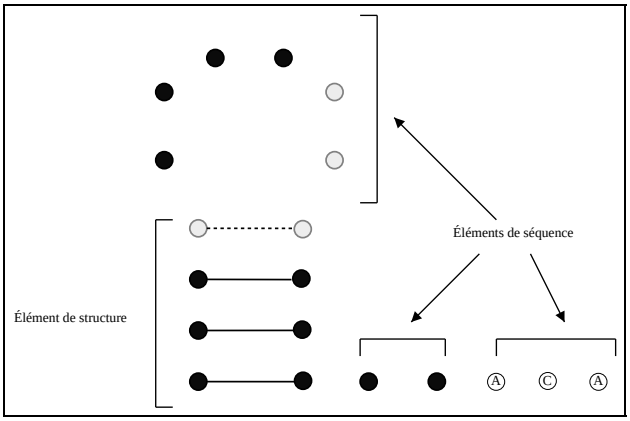


FIG. 3.1 – *Motif structuré composé d'une hélice et de trois éléments de séquence. Un cercle noir modélise une base quelconque, un cercle grisé une base quelconque pouvant être absente, et une lettre entouré d'un cercle représente une base invariante. Les traits en pointillés représentent les appariements optionnels entre bases, les traits pleins représentent les appariements obligatoires.*

3.1.2 Logiciels spécifiques ou généralistes

L'approche classique de la recherche de motifs structurés consiste à écrire un programme dédié pour chaque molécule, programme qui va identifier successivement les différents éléments de la structure. Il existe d'excellents programmes pour trouver les ARNt ou les introns de type I, par exemple. Le problème de ce type d'approche est le manque de souplesse et d'interactivité. En effet, chaque nouvelle structure à rechercher nécessite l'écriture d'un programme complet, comportant d'importantes parties de code consacrées à des tâches telles que la lecture des séquences, le déplacement à l'intérieur de la séquence, etc. Modifier un tel programme, par exemple pour tenir compte des progrès de la connaissance d'un motif, demande beaucoup de précautions, car un changement sur un paramètre peut avoir des conséquences en n'importe quel point du programme. Il n'est donc pas simple, pour un biologiste qui travaille sur une structure, de faire évoluer un programme, ou de tester directement ses hypothèses.

Les logiciels que nous avons sélectionnés dans le cadre de cette étude sont des outils classiquement utilisés soit dans le cadre de l'annotation des génomes pour rechercher des gènes appartenant à des familles spécifiques (ARNt, snoARN) soit pour la recherche de motifs.

Les logiciels spécifiques sont dédiés à une famille de molécules. Ils implémentent dans un même algorithme la description du motif recherché et l'algorithme de recherche. Il s'agit d'outils qui n'offrent pas la possibilité aux utilisateurs de décrire de nouvelles structures. Par contre ils intègrent des spécificités de la famille impossibles à exprimer avec les outils généralistes. Par exemple, CITRON [LDM94], dédié à la recherche d'introns de groupe I, incorpore la notion de motif alternatif. Snoscan [LE99], dédié à la recherche des petits ARN nucléolaires, incorpore la recherche des éléments optionnels et d'interactions avec des séquences cibles fournies dans un fichier à spécifier. Concernant la famille des ARNt, de nombreux outils spécifiques ont été développés. Ceux que nous avons utilisés sont relativement récents : tRNAscan-SE [LE97], FAStRNA [EML96] et Aragorn [LC04]. Concernant la recherche de snoARN, l'outil que nous avons utilisé est Snoscan, seul logiciel, à notre connaissance, dédié à la localisation de snoARN dans les séquences de génomes eucaryotes.

Les logiciels généralistes offrent un langage permettant à un utilisateur (familier du langage) de construire et modifier un descripteur d'une famille de molécules. Ce type d'outils permet d'affiner rapidement une recherche. Selon les outils existants, les langages proposés permettent de décrire plus ou moins finement un motif structuré. Nous avons utilisé dans notre étude les logiciels RnaMot [GMC90], RnaBob [Edd96], PatScan [DLO97], Palingol [BKV96], RnaMotif [MEG+01], COVE [ED94] et Erpin[GL01].

3.2 Étude des logiciels généralistes dans le cadre CSP

Qu'ils soient de nature spécifique ou bien généraliste, les outils que nous avons étudiés traitent la recherche de motifs structurés comme un problème de satisfaction de contraintes. En effet, tous ces outils cherchent à satisfaire un ensemble de contraintes sur un ensemble d'éléments. Afin d'une part de sortir du cadre classique des langages et grammaires et d'autre part parce que la formalisation de ce problème dans le cadre formel des CSP semble naturelle, qu'elle permet d'offrir un cadre commun de représentation à cette étude et qu'elle laisse une grande ouverture quant à l'expression et au traitement des contraintes, nous avons choisi de formaliser le problème de la recherche de motifs structurés dans le cadre CSP. Une démarche similaire a déjà été proposée par I. Eidhammer *et al* [EGJR01] mais reste limitée à certains motifs (tige-boucle).

3.2.1 Présentation du formalisme CSP

Un motif structuré peut être représenté par un ensemble d'objets connectés entre eux par des contraintes.

Le formalisme CSP offre un cadre formel de représentation et de résolution des problèmes de satisfaction de contraintes dans lequel un problème s'exprime classiquement sous la forme $P = (X, D, C)$:

- $X = x_1, ...x_N$ désigne l'ensemble des variables représentant les caractéristiques d'un ou plusieurs objets.
- $D = d_1, ..., d_N$ désigne l'ensemble des domaines associés à chaque variable et donne l'ensemble des valeurs possibles pour ces variables.

– $C = c_1, ...c_k$ désigne l'ensemble des contraintes où chaque contrainte est définie par un sous-ensemble du produit cartésien des domaines des variables impliquées dans la contrainte.

Une solution d'un CSP est une affectation de l'ensemble des variables qui satisfait l'ensemble des contraintes. Les algorithmes de recherche associés à ce formalisme développent un arbre de recherche et tirent leur efficacité de l'exploitation d'une part de la structure du CSP (la représentation sous forme de graphe du problème), et d'autre part de la propagation des contraintes avant d'entamer le développement de l'arbre de recherche.

Ce cadre formel, relativement simple et classique, semble particulièrement adapté pour la représentation et la résolution du problème de la recherche de motifs structurés. Nous verrons ci-dessous comment les différents outils existants s'inscrivent ou peuvent s'inscrire dans ce cadre. Nous évoquerons aussi les limitations du cadre classique pour intégrer des spécificités absentes actuellement de l'ensemble des outils généralistes existants.

3.2.2 Cadre de comparaison des logiciels

Nous nous sommes intéressés à l'ensemble de ces logiciels dans le but d'explorer les possibilités et les limites de leurs langages de description. Dans leur majorité, les langages permettant de décrire un motif offrent à l'utilisateur la possibilité d'énumérer un ensemble d'éléments de séquence et de structure constituant le motif. Chaque élément peut être lui même décrit plus précisément (par son contenu, sa taille...) ou par ses relations avec d'autres éléments. Plus formellement, nous appellerons :

– Variables : les éléments de référence fournis par les langages pour décrire les motifs structurés.
– Contraintes : les propriétés que doivent respecter ces éléments ou bien les relations qu'entretiennent certains éléments avec d'autres. Selon le nombre d'éléments ou variables impliqués, nous parlerons de contraintes unaires (un seul élément), binaires (deux éléments) ou k-aires (k éléments).

Les domaines de valeurs pour l'ensemble des logiciels sont représentés par l'intervalle de valeurs $[1, p]$, avec $p =$ taille de la séquence où est recherché le motif. Ainsi, les valeurs prises par les variables constituent les positions des motifs sur la séquence.

Nous nous sommes également intéressés à la méthode adoptée par les logiciels pour rechercher les solutions, et le cas échéant aux méthodes proposées pour effectuer un tri ou classement des candidats obtenus.

3.2.3 Langages utilisateur : variables et contraintes

Avant de rentrer plus précisément dans la description de chacun de ces logiciels, nous résumons avec les tableaux 3.1 et 3.2 ci-dessous les différences et similarités entre les éléments de référence (ou variables) offerts dans les langages de description proposés par les différents logiciels. Les tableaux 3.1 et 3.2 s'appuient sur le formalisme CSP présenté ci-dessus. Ils illustrent, selon un point de vue informatique, les possibilités de mise en relation entre les variables (éléments de référence proposés par chaque langage) et les contraintes.

	Rnamot	Rnabob	Patscan	Palingol	Rnamotif	Erpin	Cove
(a) Type de variables							
Élément de séquences	+	+	+	-	+	+	+
Élément de structure corrélation = 2	Hélice	Hélice Palindrome	Hélice Palindrome, Répétition	Hélice	Hélice	Hélice	
Élément de structure corrélation >2	-	-	-	-	Triple-hélice Quadruple-hélice	-	-
(b) Contraintes unaires sur les éléments de séquence							
Longueur	+	+	+	*	+	+	+
Contenu	1-3-4	1-3-4	1-3-4-5	*	1-3-4-5	2	2
Composition	+	+	+	*	+	+	+
(c) Contraintes unaires sur les éléments de structure							
Longueur	+	+	+	+	+	+	+
Contenu	+	+	+	+	+	-	-
Composition	-	-	-	-	+	-	-
Nature	-	-	+	+	+	-	-
Énergie	-	-	-	-	+	-	-
Erreurs	Mésapp.	Mésapp.	Mésapp.,Ins.,Dél.	Mésapp.	Mésapp.	-	-

TAB. 3.1 – *Types de variables (a) disponibles selon les différents langages des outils généralistes, et des contraintes unaires se rapportant aux variables de type éléments de séquence (b) ou structure (c). (+) : signifie que la contrainte/variable existe dans le langage et (-) qu'il/elle n'existe pas. (Mésap.) : Mésappariement(s) autorisés. (Ins. Dél.) : Insertion(s) et délétion(s) autorisées. Contrainte sur le contenu : (1): Mot, (2) : Alignement, (3): Consensus, (4): Expression régulière, (5): Matrice de poids/profil. (*) : Les éléments de séquence n'existent pas en tant qu'éléments de référence dans Palingol . Cependant, il est possible d'utiliser des fonctions de recherche de mots. Le résultat peut être ensuite mémorisé dans une variable dynamique, afin de pouvoir poser des contraintes dessus.*

	RnaMot	Rnabob	PatScan	Palingol	RnaMotif
(d) Contraintes binaires et k-aires entre les éléments de séquence					
Ordre	Implicite	Implicite	Implicite	*	Implicite
Distance	Déduite	Déduite	Déduite	*	Section score
Longueur	-	-	+	*	+
Autres	-	-	-	*	Section score
(e) Contraintes binaires et k-aires entre les éléments de structure					
Ordre	Implicite	Implicite	Implicite	Implicite	Implicite
Distance	Déduite	Déduite	Déduite	Déduite Section span	Déduite
Longueur	-	-	+	Section CROSS Section hélice	Section score
Autres	Nombre global: mésapp. GU-UG	-	-	Section CROSS	Section score
(f) Contraintes binaires et k-aires entre les éléments de séquence et de structure					
Distance	Déduite	Déduite	Déduite	Section CROSS	Déduite
Autres	-	-	-	Section score	Section score

TAB. 3.2 – *Contraintes k-aires se rapportant aux variables de type éléments de séquence ou de structure. (+) : la contrainte est disponible dans le langage et (-) : la contrainte est absente. Les éléments de séquence n'existent pas en tant qu'éléments de référence dans Palingol. (Nombre global: Mésap. ou GU-UG) : le nombre global de mésappariements et d'appariements de type Wobble est paramétrable. La section Score de Rnamotif et la section CROSS de Palingol permettent de spécifier de nouvelles contraintes pour évaluer les candidats trouvés par l'algorithme de recherche.*

Le tableau 3.3 est une extrapolation des tableaux précédents en énumérant les objets biologiques qu'il est possible de représenter à partir du langage de chaque logiciel, en mettant à profit les caractéristiques de chacun.

	Rnamot	Rnabob	Patscan	Palingol	Rnamotif	Erpin	Cove
Éléments biologiques modélisables							
mot	+	+	+	+	+	+	+
hélice	+	+	+	+	+	+	+
pseudo-noeud	+	+	+	+	+	+	+
répétition	-	+	+	-	-	-	-
triple-hélice	-	-	+	+	+	-	-
quadruple-hélice	-	-	+	-	+	-	-

TAB. 3.3 – *Possibilité d'expression en terme d'objets biologiques pour chaque langage.*

3.2.4 PatScan

Le langage proposé par **Patscan** est très simple mais à la fois très complet. Un motif peut être décrit par des éléments de séquence, des hélices, des triples ou quadruple hélices, mais aussi des répétitions, des palindromes et des pseudo-noeuds.

3.2.4.1 Les variables

Le descripteur définit le CSP à résoudre. Un motif structuré selon PatScan est une séquence d'unités de pattern (les variables), lesquelles peuvent être exprimées par des mots (caractères que l'on souhaite spécifier pour l'élément de séquence) ou par des intervalles de longueurs. Tout élément du motif ayant une relation avec un autre élément peut être nommé afin d'identifier les éléments en relation. C'est ainsi que sont définis les éléments autres que les séquences.

L'utilisateur peut aussi spécifier des motifs alternatifs, comme par exemple proposer deux mots possibles pouvant être trouvés à une certaine position. Si le premier mot proposé n'est pas trouvé, alors le programme essaie de trouver le second.

3.2.4.2 Les contraintes

Parmi les contraintes remarquables spécifiques à Patscan, on peut souligner :

– les contraintes permettant d'autoriser des erreurs de type mismatch/insertion/deletion aussi bien dans les éléments de séquence que dans les éléments de structure, avec pour conséquence, dans ce dernier cas, la possibilité d'autoriser la recherche d'hélices incluant un (ou plusieurs) renflement(s).

– la possibilité de spécifier des longueurs ajoutées de deux ou plusieurs éléments de séquence.

3.2.4.3 L'algorithme de recherche

L'algorithme recherche les éléments d'après l'ordre topologique du descripteur (ex: $p_1 p_2 p_3 \ldots$). La première tentative consiste à regarder si le pattern p_1 peut se positionner à la position courante, et en cas de succès à passer au pattern suivant. Les contraintes de distance sont implicites, et la descente dans l'arbre de recherche se fait en traitant les variables les unes après les autres. La variable courante est recherchée dans la région délimitée par l'occurrence de la variable précédemment traitée et la contrainte de longueur de la variable courante.

3.2.4.4 Classement des solutions

Le programme ne propose aucune procédure pour classer ou regrouper les solutions.

3.2.4.5 Exemple

Un exemple de descripteur correspondant au motif structuré de la figure 3.1 est donné par :

p1=3...4 4...6 ∼**p1** 2...2 ACA

L'élément de structure y est représenté en gras, et les éléments de séquence correspondent soit à des intervalles de taille soit à une succession de bases déterminant le contenu d'un mot.

3.2.5 RnaMot

RnaMot est le premier outil généraliste ayant permis une description intégrant aussi bien les éléments de séquence que les éléments de structure (hélices et pseudo-noeuds).

3.2.5.1 Les variables

Dans RnaMot, un descripteur va définir le CSP à résoudre. Il définit l'ensemble des variables par la liste des éléments du motif (appelés SE) sur une première ligne. Les éléments décrits sont de 2 types : les éléments de séquence (s1,s2...) et les éléments de structure ((H1,H1),(H2,H2)...). Des caractères différents sont utilisés pour chaque type d'élément afin de les différencier.

3.2.5.2 Les contraintes

L'ordre des éléments dans le descripteur ainsi que les spécifications associées par la suite à chaque élément définissent de manière implicite ou explicite l'ensemble des contraintes. Les contraintes explicites portent sur les éléments. En effet, chaque variable de la première ligne est reprise dans une ligne et ses caractéristiques sont alors spécifiées. Pour les hélices, une contrainte implicite est donnée par la répétition à une position lors de la définition des variables. Cette contrainte permettra lors de la résolution du problème de calculer la taille du simple brin (boucle) entre les deux régions appariées (cette information n'est pas fixée par l'utilisateur comme une contrainte). Chaque élément de structure (H_i) est répété deux fois afin de donner la position relative des deux brins à l'intérieur du motif. La liste des contraintes unaires est construite en reprenant chaque variable du problème et en décrivant les caractéristiques que l'élément associé à la variable doit respecter. Les contraintes binaires portent sur l'ordre des éléments entre eux et sont données par la définition des variables.

Contraintes unaires sur les éléments de séquence

On peut souligner ici la prise en compte des erreurs. En effet, cette contrainte n'est pas explicitement donnée mais une erreur peut être positionnée en mettant une lettre minuscule à une position donné d'un élément de séquence. On peut donc spécifier de manière optionnelle le contenu d'un élément de séquence en mettant en minuscule la chaîne de caractères qui définit son contenu. Par contre la contrainte sur l'existence de l'élément en tant que chaîne de caractères persiste. Les auteurs utilisent cette contrainte pour calculer un score.

Contraintes unaires sur les éléments de structure

On peut remarquer ici qu'il est possible de borner, localement, un nombre d'erreurs autorisées sur un élément de structure.

Contraintes k-aires sur les éléments de structure

La possibilité de limiter globalement le nombre maximum de mésappariements et le nombre de d'appariements de type Wobble mérite ici d'être soulignée (nous en verrons l'utilité, par exemple, dans le cas de la recherche d'ARNt).

3.2.5.3 L'algorithme de recherche

L'algorithme de recherche tient compte du fait qu'il serait très coûteux de rechercher *a priori* toutes les hélices sans considération des contraintes imposées.

L'algorithme de recherche développe un arbre de recherche (backtraking) en générant uniquement lorsque c'est nécessaire les valeurs des variables et en examinant de manière dynamique l'ensemble des contraintes. Il s'appuie sur une structure de données qui s'apparente très bien au cadre CSP. La liste des variables est apparentée à la liste des motifs . A chaque variable est associé un domaine de valeurs possibles (appelé `DOMAIN`) qui est défini par une liste d'un (pour les régions simple brin) ou deux intervalles (pour les hélices). Chaque intervalle est défini par deux entiers. Toute solution trouvée est stockée dans un tableau d'intervalles appelé `GRID`. L'ensemble des solutions est donné par une liste de `GRID`. La recherche peut se faire selon un ordre spécifié par l'utilisateur, ou selon un ordre calculé par le programme lui-même. Les performances de l'algorithme sont sensibles à cet ordre.

3.2.5.4 Classement des solutions

Le nombre de solutions pouvant être très important, RnaMot les classe dans deux fichiers en s'appuyant sur une fonction d'évaluation du motif. Il privilégie par ordre décroissant :

- Les meilleurs candidats avec des éléments de séquence qui correspondent parfaitement à la chaîne de caractères spécifiée (dans le cas où certains des caractères sont optionnels).
- Les hélices de taille maximale.
- L'énergie libre minimum sur les hélices (calculée sur la base des paramètres de Turner [TS88]). Le meilleur candidat peut ainsi être sélectionné lorsque plusieurs solutions commencent et se terminent à la même position sur la séquence. Il est alors stocké et présenté dans le fichier des meilleures solutions (extension `.sol`) alors que les autres solutions sont sauvegardées dans un fichier appelé alternatif (extension `.alt`).

3.2.5.5 Exemple

Un exemple de descripteur correspondant au motif structuré de la figure 3.1 est donné :

```
H1 s1 H1 s2 s3
H1 3:4
s1 4:6
s2 2:2
s3 3:3 ACA
```

3.2.6 RnaBob

RnaBob propose un langage assez proche de celui de RnaMot. Il est ainsi possible de définir un motif contenant des éléments de séquence et de structure impliquant des interactions entre deux régions d'une même molécule. Les descriptions d'hélices, de palindromes, de répétitions et de pseudo-noeuds sont possibles.

3.2.6.1 Les variables

Dans RnaBob, un descripteur va définir le CSP à résoudre. La localisation des différents éléments du motif est décrite sur une première ligne, à l'instar du langage de RnaMot. Les

éléments décrits sont de deux types : les éléments de séquence (s1,s2...) et les éléments de structure ((H1,H1'),(H2,H2')...).

3.2.6.2 Les contraintes

Les particularités du descripteur ressemblent en grande partie à celles de RnaMot. La principale différence est liée à la façon d'autoriser les mésappariements, plus souple dans RnaBob.

Contraintes unaires sur les éléments de séquence

Ces contraintes sont classiques, à l'instar de RnaMot.

Contraintes unaires sur les éléments de structure

Ces contraintes sont aussi classiques, à l'instar de RnaMot.

3.2.6.3 L'algorithme de recherche

L'algorithme de recherche est un automate (section 4.1.4, 134) fini non-déterministe très similaire aux algorithmes de *pattern-matching* utilisés par UNIX pour les expressions régulières. La différence principale concerne la construction dynamique de certains états, fonction des occurences trouvées pour des états précédents. Cette modification permet ainsi de rechercher des complémentarité.

3.2.6.4 Classement des solutions

Le programme ne propose aucune procédure pour classer ou regrouper les solutions selon un critère.

3.2.6.5 Exemple

Un exemple de descripteur correspondant au motif structuré de la figure 3.1 est:

```
H1 s1 H1' s2 s3
H1 0:0 *NNN:NNN*
s1 0 NNNN**
s2 0 NN
s3 0 ACA
```

3.2.7 Palingol

Palingol a été le premier outil à offrir un langage déclaratif suffisamment riche pour exprimer des contraintes de nature différente.

La philosophie de Palingol diffère des logiciels présentés précédemment pour lesquelles la recherche des hélices du motif structuré est effectuée au fur et à mesure de la recherche du motif dans sa globalité. En effet, le moteur de recherche de Palingol ne travaille pas directement sur les séquences mais sur la liste calculée auparavant, de tous ses palindromes, qui sont autant d'hélices potentielles. Palingol est proposé par les auteurs avec un programme de recherche de palindromes, nommé Palamou car il permet de tenir compte de n'importe quel type d'appariements entre bases.

3.2.7.1 Les variables

Dans Palingol, un seul type de variable existe : les variables de type hélice. Une hélice est composée de 3 éléments (région 5': *head*, région 3': *tail* et boucle: *loop* de l'hélice) à partir desquels différentes contraintes unaires proposées par le langage ou bien programmées par l'utilisateur pourront être spécifiées. Les variables sont décrites de façon ordonnée selon l'apparition de leur région 5'.

Pour chaque variable de type hélice, le langage permet de spécifier des caractéristiques sur cette variable. L'ordre des variables est important car des contraintes k-aires (impliquant plusieurs variables) peuvent être définies entre des variables de type hélice à partir d'une des autres variables et parce qu'une variable dynamique définie préalablement et impliquant une autre variable hélice doit pouvoir " voir " l'instance courante de la variable.

Les contraintes unaires sur les hélices ont deux origines. La première origine vient du programme Palamou qui permet de calculer, à partir de "modèles" d'hélices, la liste des valeurs possibles pour l'ensemble des variables hélice qui seront définies dans le programme Palingol. La deuxième origine vient de Palingol. Chaque variable est à nouveau définie et un deuxième ensemble de contraintes peut alors être spécifié pour chacune des variables. C'est aussi essentiellement au moyen de ces contraintes qui vont porter sur les trois éléments associés à l'hélice, que l'on va pouvoir spécifier l'existence d'éléments de séquence.

3.2.7.2 Les contraintes

Contraintes unaires sur les éléments de structure avec Palamou

Avec Palamou, l'utilisateur peut spécifier la taille de la boucle d'une hélice, son énergie minimum. Les autres contraintes sont classiques. Il nous semble aussi utile de préciser que les mésappariements sont autorisées dans une hélice mais seulement à l'intérieur de cette dernière. Les mésappariements aux extrémités ne sont pas acceptés. Ceci peut expliquer, dans certains cas des différences entre résultats, comme nous le verrons dans le cadre de la recherche des ARNt.

Contraintes unaires sur les éléments de structure avec Palingol

Chaque hélice est en fait considérée comme une tige-boucle et contient 3 éléments qui sont sa région 5', sa région 3' et la région simple brin qui les sépare.

- Les contraintes peuvent porter sur chaque attribut d'un élément de l'hélice. Les attributs permettent d'accéder aux positions de début et de fin de chaque élément, au contenu de chaque élément, à la longueur de chaque élément...
- Les contraintes utilisées sur ces variables sont des opérateurs de plusieurs types: booléens, numériques...
- En dehors des variables hélices, qui doivent impérativement exister, tous les autres motifs spécifiés par des contraintes relativement aux variables hélices peuvent être alternatifs par combinaison des différents opérateurs y compris les opérateurs booléens.

Contraintes binaires et k-aires sur les éléments de structure

Les contraintes binaires concernent les distances entre les différentes hélices. Elles font l'objet d'un ensemble particulier de contraintes qui sont définies dans une section dite CROSS. Comme les variables associées aux hélices sont ordonnées dans leur description selon leur ordre

d'apparition dans le motif, une sous-partie de CROSS (appelée SPAN) permet de définir des distances entre elles, les positions faisant référence par défaut étant les premières positions des régions en 5'.

La section SPAN permet ainsi de fixer une distance maximale entre chaque couple d'hélices adjacentes dans le descripteur afin de réduire l'espace de recherche. En effet, si le descripteur contient deux hélices séparées par au maximum l bases il sera inutile de considérer un couple d'hélices de la séquence courante distants de $l + k$ bases (k >0).

La section CROSS permet d'indiquer les contraintes globales , c'est-à-dire les relations entre deux (ou plus) hélices, par exemple leurs positions relatives, ou bien des conditions plus complexes telles que "les quatre premières bases de la boucle de l'hélice 1 sont complémentaires à quatre bases quelconques de la boucle de l'hélice 2".

3.2.7.3 L'algorithme de recherche

L'algorithme de recherche ne tient pas compte du fait qu'il est très coûteux de rechercher *a priori* toutes les hélices sans considération des contraintes imposées. L'handicap principal en terme d'efficacité est lié au nombre important d'hélices calculées par palamou (la taille du fichier à partir du descripteur d_1 initial sur le génome de *E. coli* est de 500 Mo).

L'algorithme de recherche est un algorithme de type *branch and bound*. Palingol sélectionne un palindrome candidat pour être l'hélice 1, et vérifie une à une les contraintes sur cette hélice (contraintes de type helix). Dès qu'une contrainte n'est pas vérifiée, il s'arrête et prend le palindrome suivant. Quand il a trouvé un palindrome susceptible d'être l'hélice 1, il prend le suivant comme candidat pour l'hélice 2, vérifie les contraintes, et tant qu'elles ne sont pas toutes vérifiées, passe au suivant (dans la limite de la contrainte de SPAN), jusqu'à avoir toutes les hélices nécessaires. Ensuite, il vérifie les contraintes de type cross sur cet ensemble d'hélices, et à nouveau, s'arrête dès qu'une contrainte n'est pas vérifiée.

3.2.7.4 Classement des solutions

Palingol ne propose aucun outil automatique de classement des solutions. Cependant, l'utilisateur qui souhaite sélectionner des solutions peut le faire lui-même en créant son propre système de score et en évaluant d'une part chaque élément du motif, d'autre part le motif globalement.

3.2.7.5 Exemple

Un exemple de descripteur correspondant à la description de l'hélice de la figure 3.1 est :

Section helix:
(ge loop 4)
(le loop 6)
(ge head 3)
(le head 4)
(strstr (sstr (seq fullseq) (add (end tail) 2) 3) "AAC")

3.2.8 RnaMotif

Le langage de RnaMotif est aussi très riche et adapté à la spécification de motifs d'ARN. Il autorise la description de motifs contenant des éléments de séquence et de structure impliquant des interactions jusqu'à 4 régions d'une même molécule. On peut donc décrire des hélices, des pseudo-noeuds, des triples et quadruples hélices.

3.2.8.1 Les variables

Dans RnaMotif, un descripteur va définir le CSP à résoudre. Il définit l'ensemble des variables par la liste des éléments du motif. Les éléments décrits sont de 2 types : les éléments de séquence (ss) et les éléments de structure (h5,h3). Des caractères différents sont utilisés pour chaque type d'élément afin de les différencier.

3.2.8.2 Les contraintes

Les contraintes unaires sur les éléments de séquence

On peut souligner ici la richesse du langage qui offre la possibilité de traiter les mésappariements et leur localisation. Par exemple, la nature du contenu à une position donnée peut être fixée dans la section SITES.

Les contraintes unaires sur les éléments de structure

Parmi les possibilités offertes par le langage, la possibilité d'imposer la nature des appariements localement ou globalement pour l'ensemble des hélices peut être dans certains cas d'une grande utilité. De même, la possibilité d'agir sur la composition (via le nombre ou le pourcentage de mésappariements autorisés) est une particularité très intéressante.

Contraintes k-aires

Ces contraintes sont essentiellement contenues dans la section score. Différents opérateurs permettent de poser des contraintes classiques (booléens, arithmétiques, chaînes de caractères). Ces opérateurs sont sensiblement les mêmes que pour Palingol (décrit précédemment) et peuvent porter sur les mêmes types d'éléments (région 5' de l'hélice, région 3', boucle...). Seulement à cet endroit peuvent être définies de nouvelles variables. Ceci permet d'éviter les "effets de bord" potentiellement présents dans Palingol.

3.2.8.3 L'algorithme de recherche

Après avoir testé les spécifications du motif pour vérifier la cohérence entre les différentes contraintes posées, l'algorithme de recherche va développer un arbre de recherche en profondeur d'abord. Cet arbre de recherche est construit en tenant compte de la topologie du motif recherché. A ce stade de la recherche, aucune contrainte k-aire n'est testée. Ce n'est que lorsqu'un élément n'est pas trouvé que l'algorithme fait un retour arrière. Si tous les éléments du motif existent, alors les contraintes (k-aires) de la section score sont vérifiées. La solution est acceptée ou rejetée.

3.2.8.4 Classement des solutions

Une procédure spécifique permet d'éliminer des solutions contenues dans d'autres, ce qui permet de réduire le nombre de solutions à examiner. Un classement peut aussi être réalisé en utilisant l'une des deux fonctions qui sont proposées (énergie libre, complexité de la composition) ou bien en se construisant sa propre fonction de score.

3.2.8.5 Exemple

Un exemple de descripteur correspondant au motif structuré de la figure 3.1 est :

```
descr
      h5(minen=3,maxlen=4)
            ss(minlen=4, maxlen=6)
      ss( len=2)
      ss( len=3, seq="aca" )
```

3.2.9 Cove

Un modèle de covariance constitue une approche générale à plusieurs problèmes de biologie moléculaire: la prédiction de structures secondaires, l'alignement multiple de séquences, la recherche de similarités et la prédiction de séquences dans une base de données.

Les modèles de covariance ont permis une nouvelle approche de plusieurs problèmes d'analyse des ARN. Ils permettent de prendre en compte statistiquement la structure primaire et secondaire des ARN.
Un modèle probabiliste caractérisant la famille analysée est calculé à partir d'un ensemble de séquences de cette famille (alignées ou non).

Aucun langage utilisateur n'est disponible pour l'utilisateur qui souhaiterait préciser des éléments de structure.

Le modèle de covariance d'Eddy et Durbin [ED94] a été utilisé pour la première fois sur les ARNt. Il constitue à ce jour l'outil le plus fiable pour localiser des ARNt dans les séquences génomiques.

Cependant, l'algorithme de recherche reste en pratique très limité par la taille des séquences qu'il peut traiter (pas plus de 200 nucléotides).

3.2.10 Erpin

L'approche d'Erpin est basée sur des profils de structure secondaire calculés à partir de séquences dites "d'apprentissage". Deux types de matrices de poids sont construites à partir d'un alignement :

(i) les éléments de séquence sont représentés par un profil et

(ii) le modèle calculé à partir des éléments de structure est représenté par un "profil d'hélice étendu" comprenant 16 *lod-scores* par position (correspondants aux 16 appariements possibles) [GL01].

Erpin prend en compte des informations de séquence primaire et secondaire.

Quatre types de structures élémentaires peuvent être identifiées, à partir de l'alignement fourni : les tige-boucles (une hélice et une boucle), les hélices (sans boucle), les mots, et les combinaisons de deux hélices imbriquées ou non (les pseudo-noeuds sont possibles). Les structures plus complexes sont décomposées à partir de ces structures de référence et leur recherche consiste à combiner la recherche de ces structures de référence. **L'algorithme de recherche**

Pour rechercher une tige-boucle, l'algorithme positionne à la position i le brin 5' de l'hélice. Le brin 3' de cette même hélice pouvant se trouver à une distance variable (les brèches dans l'alignement de référence sont prises en compte seulement dans les régions simple brin), l'algorithme recherche la meilleure position. Pour cela, il calcule deux scores pour l'ensemble des positions possibles : S_h correspondant à l'alignement de $H5$ en position i et de $H3$ en position j avec le profil pré-calculé, et S_s le score obtenu par programmation dynamique entre la région simple brin délimitée par la position de $H3$ et le profil pré-calculé. L'algorithme choisit la position de $H3$ en maximisant la somme de ces deux scores, et si la valeur est supérieure au seuil paramétré, il retient comme candidat la solution ainsi trouvée.

Erpin permet à l'utilisateur de spécifier en ligne de commande une région pour la construction de son modèle et de cibler ensuite la recherche sur des éléments particuliers (mot ou hélice par l'utilisation de masques). De plus, par le jeu de ces masques il est aussi possible d'effectuer une recherche en plusieurs passages (le premier passage peut s'intéresser aux motifs les plus discriminants par exemple) pour en améliorer la vitesse d'exécution. Un système de score est également proposé pour sélectionner certaines prédictions.

3.3 Cadre de la comparaison des logiciels

Nous présentons dans cette section les résultats obtenus avec les outils généralistes (présentés ci-dessus) et spécifiques de recherche de motifs, lors de la recherche de deux types de motifs structurés: les petits ARN nucléolaires (snoARN) à boîtes C/D et les ARN de transfert (ARNt). Selon la molécule recherchée, notre démarche a été différente.

En effet, dans le cas particulier des ARN de transfert, un certain nombre de logiciels spécifiques utilisant différentes approches existent et donnent de très bons résultats. Nous avons cependant (dans le cadre d'une analyse parallèle) choisi cette molécule, dont la structure secondaire est bien connue, pour étudier l'ensemble des logiciels généralistes auxquels nous nous intéressons.

3.3.1 Définitions

Pour effectuer nos évaluations (outils spécifiques principalement) nous avons, pour chaque programme, comptabilisé le nombre de vrais positifs, faux positifs et faux négatifs :

Vrais positifs (VP) : candidats qui sont effectivement des gènes appartenant à la famille recherchée.

Faux positifs (FP) : candidats qui ne sont pas des gènes appartenant à la famille recherchée.

Faux négatifs (FN) : ceux qui sont des gènes appartenant à la famille recherchée et qui ne sont pas trouvés.

Ces comptages permettent de définir les deux notions suivantes :

Sensibilité (SE) : un programme est d'autant plus sensible qu'il va proposer des vrais positifs.

$$SE = VP/(VP + FN)$$

Spécificité (SP) : un programme est d'autant plus spécifique qu'il ne va pas proposer des faux positifs.

$$SP = VP/(VP + FP)$$

3.3.2 Données utilisées

Les jeux de données qui ont été utilisés sont de deux types :

– banques de données génomique [17]

– banques de données génomiques d'ARNt [18]

3.3.2.1 Les ARN de transfert

L'ensemble des logiciels généralistes pour la recherche de motifs met à la disposition de l'utilisateur un langage spécifique qui lui permet de spécifier plus ou moins facilement l'ensemble des contraintes modélisant le motif qui l'intéresse. Concernant les ARNt, nous avons ainsi essayé de comparer l'efficacité des outils sur la base des résultats attendus , mais aussi en regardant leur exhaustivité et leur vitesse d'exécution. Nous avons mené parallèlement deux types d'analyses, l'une étant liée aux logiciels généralistes avec l'objectif d'analyser l'expressivité des langages proposés et l'autre aux logiciels spécifiques avec l'objectif d'apprécier la sensibilité et spécificité des résultats obtenus.

Logiciels généralistes

Nous avons étudié les possibilités en terme d'expressivité des différents langages de description proposés par chaque logiciel à un utilisateur. A partir d'une description fournie dans la littérature, nous avons essayé de la traduire dans chaque langage. L'objectif de ce travail est d'explorer à travers l'exemple d'un descripteur donné les possibilités, différences et ressemblances entre les outils disponibles. Notre propos n'a pas porté sur l'analyse des descripteurs avec l'idée de les améliorer et d'obtenir ainsi une meilleure spécificité ou sensibilité. Cependant, nous abordons cette possibilité dans la discussion afin de fournir au lecteur quelques idées des possibilités de certains langages.

Le point de départ de notre investigation a été d'analyser les possibilités des différentes représentations et de relever un ensemble de points qui nous paraissent intéressants ou essentiels. Ces points portent sur la possibilité de décrire notamment :

– des éléments de séquence (mots) et de structure secondaire (hélices) de base non exacts ;

– des éléments structurés tels que triple hélice, quadruple hélice ou pseudo-noeud (la plupart des logiciels généralistes);

– des contraintes sur le contenu d'un élément (mot ou hélice) ;

17. URL : http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Genome

18. URL : http://rna.wustl.edu/GtRDB/

- la pondération d'un élément ou d'un ensemble d'éléments de motif dans le cadre d'une fonction de score globale ;
- des contraintes de distance ;
- des règles d'appariement et le contrôle du nombre d'appariements d'un type donné.

Dans le cadre de la première analyse liée à la recherche d'ARNt avec des outils généralistes, le comptage des vrais positifs, faux positifs et faux négatifs est donné à titre indicatif. En effet, nous discutons de la présence ou absence de vrais positifs afin d'analyser et d'illustrer les différences entre les nombre totaux de solutions pour différents logiciels à "descripteurs égaux". L'analyse des logiciels en terme de spécificité et de sensibilité n'a pas lieu d'être dans ce contexte. Les descripteurs utilisés modélisent de façon générale la famille d'ARNt et nous n'avons pas essayé de les améliorer.

Logiciels spécifiques

Nous nous sommes intéressés à un certain nombre de logiciels existants dédiés à la recherche d'ARNt. Contrairement à la première étude, l'objectif est de comparer les différentes approches en terme de sensibilité et spécificité entre elles. Les logiciels spécifiques utilisés sont tRNAscan-SE [LE97], FAStRNA [EML96] et Aragorn [LC04]

3.3.2.2 Les snoARN à boîtes C/D

Une seconde partie de ce chapitre porte sur la recherche de snoARN à boîtes C/D. A notre connaissance, un seul outil spécifique est disponible pour cette famille d'ARN : Snoscan [LE99]. Dans ce contexte de recherche de molécules, nous avons comparé (selon le point de vue d'un biologiste) trois approches différentes, deux ayant recours à des logiciels généralistes et une utilisant Snoscan.

3.4 Recherche d'ARN de transfert

Les ARNt jouent un rôle capital dans la biosynthèse protéique en apportant les acides aminés jusqu'à l'usine ribosome où s'effectue la synthèse protéique.

Le premier modèle auquel nous nous sommes intéressés pour notre étude comparative est l'ARN de transfert. Cette molécule présente l'avantage d'avoir une structure secondaire bien caractérisée et conservée entre espèces.

3.4.1 Description de la molécule

Les ARNt sont constitués d'un brin d'ARN replié sur lui-même, comportant 60 à 95 nucléotides dont quelques uns sont rares (dihydro-uridine, pseudo-uridine). Ils possèdent une structure secondaire (Fig. 3.2) dite en feuilles de trèfle. Ils adoptent également une structure tertiaire en forme de L qui leur permet d'inter-agir avec le ribosome. La plus grande boucle de chaque

ARNt possède un triplet de nucléotides spécifique appelé anticodon. L'anticodon peut s'associer par appariement des bases à un codon complémentaire de l'ARN messager. L'extrémité 3' porte le site d'attachement de l'acide aminé (triplet CAA).

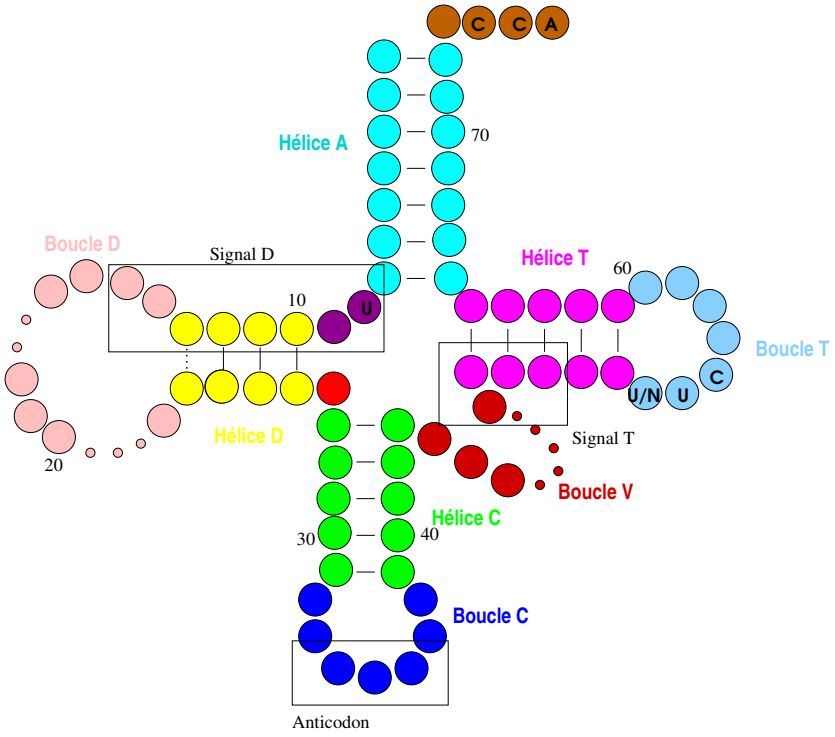


FIG. 3.2 – *Structure secondaire canonique en trèfle d'un ARN de transfert. Les rectangles représentent les deux signaux T et D considérés par les logiciels tRNAscan et FAStRNA. Un intron éventuel peut apparaître entres les positions 37 et 38.*

3.4.1.1 Structure particulière de l'ARN de transfert de la sélénocystéine

Les ARN de transfert de la sélénocystéine diffèrent de la structure canonique des ARN de transfert standards. Ils présentent un bras accepteur de 8 paires de bases, une longue région variable ainsi que quelques substitutions à plusieurs positions bien conservées parmi les ARN de transfert canoniques.

La Sélénocystéine [LGF+00] est un acide aminé présent dans plusieurs enzymes. Ce 21 iéme acide aminé a une structure similaire à celle de la cysteine mais contient un atome de sélénium à la place de l'atome de sulfure habituel. La sélénocystéine n'est pas un acide aminé habituel et est codé par le codon stop UGA.

L'ARNt nécessaire à l'encodage de cet acide aminé participe avec d'autres facteurs à la redéfinition d'un codon stop UGA. Chez les procaryotes, un codon stop UGA sera interprété comme un codon sélénocystéine si une tige-boucle est présente juste après le codon stop. Chez les mammifères, il faut également une structure secondaire nommée SECIS dans les régions 3' non traduites de l'ARNm. Grâce à un mécanisme particulier (encore mal compris mais où plusieurs facteurs interviennent), ces codons STOP sont recodés par des codons sélénocystéines.

Ces différences rendent la recherche plus difficile à partir de logiciels dédiés à la structure canonique des ARNt.

3.4.2 Les algorithmes sur mesure pour les ARNt

Les premiers algorithmes pour la recherche d'ARNt [EM96] utilisaient les informations de structure primaire et/ou secondaire. Par exemple, Staden [Sta80] a été le premier à proposer un programme pour rechercher la structure secondaire, et tester ensuite la présence des bases invariantes ou semi-invariantes. Marvel [Mar86], par la suite, a utilisé les mêmes étapes, mais en les effectuant dans le sens inverse. Shortridge *et al* [PSP86] se sont plus particulièrement intéressés à la structure secondaire, et ont proposé plusieurs programmes de recherche d'ARNt en fonction des différentes catégories d'ARNt structure dépendante (procaryotiques, eucaryotiques, mitochondriaux et avec intron). Ces programmes étaient peu efficaces en terme de temps de calcul et depuis, d'autres programmes ont été développés.

3.4.2.1 tRNAscan

tRNAscan [FB91] a été l'un des premiers programmes à proposer des résultats intéressants de manière efficace. Il recherche trois types de caractéristiques :

- Les signaux caractéristiques de régions conservées (le signal TpsiC et le signal D) par une approche probabiliste (introduite initialement par Staden [Sta89]) en utilisant des matrices de consensus. La recherche d'un signal se fait par le calcul d'un score et la région est considérée comme conforme si le score dépasse un certain seuil.
- Les hélices formant les quatre bras de la structure en trèfle caractéristique des ARNt.
- La localisation et la longueur d'un intron potentiel.

Ces différentes caractéristiques sont recherchées au cours de 7 étapes séquentielles. Un système de score permet de décider à chacune des étapes si l'on peut continuer ou considérer que l'on n'a pas de gène d'ARNt dans la région considérée.

Le programme incrémente de 1 le score général lorsqu'une étape est franchie avec les conditions idéales, de 0 si les conditions sont comprises entre la valeur seuil et maximale. Il arrête sa recherche si l'étape considérée ne dépasse par une valeur seuil. Pour qu'un ARNt soit prédit, le score général doit être au moins égal à 4 à l'étape 5, et à 5 à l'étape 7.

3.4.2.2 Pol3Scan

Contrairement à tRNAscan, Pol3scan [PCB$^+$94] recherche uniquement les éléments de séquence pour les ARN de transfert d'eucaryotes, sans tenir compte des informations de structure secondaire connues de la molécule. Il recherche les régions intragéniques de contrôle transcriptionnel (appelées boîtes A et B), le signal de terminaison de transcription de l'ARN polymérase III, et des intervalles particuliers de distance entre ces trois régions.

Chaque étape est clôturée par la comparaison du score nouvellement obtenu à un score intermédiaire, et si le score obtenu n'est pas supérieur à cette valeur, la fenêtre de recherche est décalée de 1 et la recherche est réinitialisée.

La détection particulière d'ARN de transfert de la sélénocytéine se fait à partir de l'étape 1, lorsque le score obtenu pour la recherche de la boîte B se trouve dans un intervalle défini. L'algorithme recherche ensuite la présence d'une boîte A spécifique aux sélénocystéines, d'un site de terminaison correctement positionné ....
En se basant exclusivement sur les éléments de séquence, cet outil de recherche d'ARNt eucaryotes permet de reconnaître des ARN de transfert avec une structure inhabituelle (exemple les ARNt sélénocystéine). L'inconvénient réside dans la difficulté à distinguer les gènes d'ARN de transfert avec une activité transcriptionnelle active des éléments dérivés des ARNt de transfert. Pavesi et *al* [PCB$^+$94] ont comparé les résultats obtenus avec Pol3scan et tRNAscan pour un même jeu d'essai. Ils annoncent un nombre de faux négatifs trois fois plus élevés pour tRNAscan, et un nombre de faux positifs quatre fois moins grand. Les deux algorithmes sont complémentaires dans leurs approches et leurs résultats, l'utilisation des deux permettant d'obtenir un nombre de faux négatifs proche de zéro.

3.4.2.3 tRNAscan-SE

tRNAscan-SE [LE97] est le logiciel le plus utilisé actuellement pour rechercher les ARN de transfert dans les projets génomiques. Il utilise 3 approches différentes, combinant la spécificité d'un modèle de covariance avec la rapidité et la sensibilité de deux programmes [FB91, PCB$^+$94] complémentaires : Pol3scan et tRNAscan 1.3. Trois étapes sont réalisées dans le cadre de la recherche :

Étape 1 : Une liste de candidats ARNt est recherchée à l'aide des programmes tRNAscan 1.4 (adaptation de tRNAscan 1.3 [LE97]) et EutfindtRNA (adaptation de Pol3scan [LE97]). Les deux programmes rapides ont une sensibilité en valeur ajoutée supérieure à 99% (la combinaison des deux permet de trouver presque tous les candidats attendus : vrais positifs) mais avec une spécificité totale très mauvaise (beaucoup de faux positifs).

Étape 2 : Chaque séquence d'ARNt est évaluée à l'aide du programme Cove. Le principal défaut de cette méthode est le temps de calcul nécessaire (3 jours pour le génome de *E. coli*). L'utilisation des deux programmes précédents en amont de celui-ci, permet à tRNAscan-SE d'appliquer les atouts de Cove en des temps raisonnables.

Étape 3 : Une autre fonctionnalité du programme Cove est utilisée pour replier les ARNt prédits et détecter l'anticodon qu'ils possèdent.

Bien que les ARN de transfert des organelles ont une structure secondaire et primaire moins bien conservée par rapport aux ARNt nucléaires, l'utilisation de tRNAscan-SE sur un génome mitochondrien a donné de très bon résultats. Le mode prévu pour les séquences d'organelles revient à utiliser tRNAscan-SE sans pré-filtre, avec le modèle de covariance approprié calculé à partir d'ARNt d'organelle.

3.4.2.4 FAStRNA

L'algorithme FAStRNA [EML96] suit la même démarche globale que l'algorithme de Fichant et Burks [FB91] . En effet, il s'intéresse à la fois à la séquence primaire en recherchant certaines bases invariantes ou semi-invariantes, et à la séquence secondaire en trèfle.
Le programme existe sous deux formes, **FAStRNA-CM** et **FAStRNA-CLASS**, suivant la façon dont est considéré un signal. La procédure utilisée ensuite pour rechercher la structure secondaire est identique dans les deux cas.

Recherche d'un signal selon chaque méthode

FAStRNA-CM Cette première implémentation est probabiliste. Elle utilise des matrices de fréquence. La recherche d'un signal est effectuée dans une fenêtre glissante. La première étape consiste à repérer si la fenêtre contient les bases invariantes à k erreurs près. Si tel est le cas, un score de similitude avec la matrice consensus (les bases invariantes ne sont plus prises en compte) est calculé lors de la seconde étape. Un filtre est ensuite appliqué à ce score pour retenir la séquence contenue dans la fenêtre.

FAStRNA-CLASS La seconde implémentation suit une approche de "pattern-matching" en considérant le signal comme une expression régulière (recherchée à k erreurs près). Cette méthode permet ainsi de tenir compte des fréquences en purine, pyrimidine ou de tout autre sous-ensemble de l'alphabet des nucléotides.

L'expression est construite à partir d'une matrice consensus où sont éliminés à chaque position les nucléotides présentant une fréquence inférieur à 0.05. Contrairement au premier, cet algorithme n'utilise pas de notions de seuils d'acceptation et de formule de score. Le signal est considéré dans son ensemble et n'est plus découpé en bases invariantes ou semi-invariantes.

Recherche de la structure secondaire

Contrairement à tRNAscan qui considère seulement les appariements de type Watson-Crick et Wobble, FAStRNA attribue un score à chacune des combinaisons des 4 bases. Ce score pouvant être positif ou négatif, il n'est plus nécessaire de compter le nombre d'appariements pour accepter l'hélice mais de filtrer selon un certain score et de refléter [EML96] les stabilités thermodynamiques des appariements. Par exemple, une hélice devant être composée de 5 appariements selon tRNAscan pourra être accepté par FAStRNA avec une longueur de 4 appariements si ils sont tous de type G-C (le plus stable).

Prédiction d'un ARN de transfert

L'algorithme de FAStRNA est séquentiel et la recherche d'un ARN de transfert se fait en 8 étapes.

Étapes 1-2 : Recherche de la région T composée d'une boucle conservée et d'une hélice (identique à tRNAscan).

Étape 3 : Recherche du signal D à un intervalle donné de la région T (intron potentiel pris en compte).

Étape 4 : Recherche du bras de l'amino-acyl positionné en fonction du signal D.

Étape 5 : Recherche du bras D.

Étape 6 : Première comparaison à un score global pour vérifier si la portion prédite est conforme à une structure type.

Étape 7 : Recherche du bras de l'anticodon avec insertion d'un intron en cas d'échec. La taille initiale de cet intron est de 8 nucléotides et est incrémentée de 1 nucléotide à chaque nouvelle tentative infructueuse jusqu'à arriver à une distance minimale du bras T.

Étape 8 : Comparaison du score final avec un score global pour prédire ou non un ARN de transfert. L'hypothèse sous-jacente à cette comparaison est l'hypothèse d'un mécanisme compensatoire dont le principe est que la présence d'une région de mauvaise qualité est toujours compensée par la présence d'une région de bonne qualité.

N. El Mabrouk, pour optimiser les temps d'exécution de FAStRNA, a mis au point une nouvelle méthode de recherche approchée de motifs en utilisant l'algorithme de Shift-add (algorithme de Baeza-Yates et Manber voir section 2.2.3, 54). Cette amélioration concerne la recherche d'un signal dans le cas FAStRNA-CLASS. Les résultats de cette étude constituent l'essentiel de sa thèse [EM96].

3.4.2.5 Aragorn

Aragorn [LC04] est un outil récent dédié à la localisation d'ARNt et d'ARNtm[19]. Il organise la recherche des éléments caractérisant les ARNt autour de la présence des signaux et des 4 hélices. L'algorithme commence par rechercher une sous séquence ($GTTC$ avec mésappariements mais sans insertion ni deletion) de la boîte B. A partir de chaque candidat, il cherche l'hélice T et la boucle T. En cas de succès, l'algorithme recherche en amont de cette première région une sous séquence de la boîte A, et essaie ensuite de reconstruire la structure secondaire de l'ARNt. Il intègre également des notions de structure tertiaire pour améliorer la valeur du score calculée pour chaque candidat.

3.4.3 Logiciels généralistes

Concernant les logiciels généralistes, nous avons utilisé plusieurs descripteurs d'ARNt en essayant de les traduire aussi fidèlement que possible selon les possibilités des langages de ces logiciels.

L'étude menée avec les outils généralistes n'a pas la même finalité que celle utilisant les outils spécifiques L'objectif principal est d'essayer de décrire, à l'aide de chaque langage, les mêmes informations. Par cette approche, nous avons essayé de les étudier dans un cadre rigoureusement identique.

19. L'ARN transfert-message (ARNtm), présent chez les procaryotes, possède une structure et une fonction hybrides entre l'ARNt et l'ARNm. Un des rôles de cette molécule consiste à débloquer le dispositif de traduction d'un ARNm problématique (absence de codon stop).

3.4.3.1 Les descripteurs d_0, d_1 et d_2

Nous avons testé les quatre outils généralistes avec deux descripteurs, issus de RnaMot [GMC90]. Le premier d'entre eux, d_0, est cité comme un descripteur typique d'ARNt, avec un mésappariement toléré dans l'hélice du bras accepteur et des longueurs constantes pour les quatre hélices.

Le second descripteur, d_1, plus dégénéré ([GMC90] : la sensibilité et spécificité augmente) autorise deux mésappariements dans le bras accepteur de l'amino-acide et un mésappariement dans les trois autres hélices. Ces deux premiers descripteurs seront utilisés pour rechercher les ARN de transfert procaryotiques.

A partir de tests successifs, nous avons déduit le troisième descripteur, d_2 pour modéliser les ARN de transfert eucaryotiques. Il diffère du descripteur d_1 en autorisant la présence d'un intron potentiel dans la boucle de l'anticodon (élément simple brin de taille variable) et en ne caractérisant plus le motif NCCA (nécessaire pour fixer l'amino-acide) en position 3' terminale. Chez les eucaryotes, une protéine est en charge d'ajouter ce motif aux ARN de transfert mature.

Nous avons traduit ces caractéristiques avec les langages fournis par PatScan (Tab 3.4), RnaMotif (Tab 3.7), RnaBob (Tab 3.5), RnaMot (Tab 3.6) et Palingol (Tab 3.9) avec autant de fidélité que nous le permettent les différentes syntaxes.

3.4.3.2 Erpin

Nous avons utilisé les trois lots de séquences d'apprentissage d'ARNt fournis avec le programme Erpin (Tab 3.8).

d_0
r1=AU,UA,GC,CG,GU,UG p1=7...7 UN p2=4...4 4...14 r1∼p2 1...1 p3=5...5 6...7 r1∼p3 2...22 p4=5...5 UUC 4...4 r1∼p4 r1∼p1[1,0,0] NCCA
d_1
r1=AU,UA,GC,CG,GU,UG p1=7...7 UN p2=3...4 4...14 r1∼p2[1,0,0] 1...1 p3=5...5 6...7 r1∼p3[1,0,0] 2...22 p4=5...5 UUC 0...4 r1∼p4[1,0,0] r1∼p1[2,0,0] NCCA
d_2
r1=AU,UA,GC,CG,GU,UG p1=7...7 UN p2=3...4 4...14 r1 ∼p2[1,0,0] 1...1 p3=5...5 6...60 r1 ∼p3[1,0,0] 2...22 p4=5...5 NUC 0...4 r1 ∼p4[1,0,0] r1 ∼p1[2,0,0]

TAB. 3.4 – *PatScan.*

	d_0	d_1	d_2
	H1 s1 H2 s2 H2' s3 H3 s4 H3' s5 H4 s6 H4' H1 s7		
H1	1:1 NNNNNNN:NNNNNNN	2:2 NNNNNNN:NNNNNNN	2:2 NNNNNNN:NNNNNNN
H2	0:0 NNNN:NNNN	1:1 *NNN:NNN*	1:1 *NNN:NNN*
H3	0:0 NNNNN:NNNNN	1:1 NNNNN:NNNNN	1:1NNNNN:NNNNN
H4	0:0 NNNNN:NNNNN	1:1 NNNNN:NNNNN	1:1 NNNNN:NNNNN
s1	0 UN	0 UN	0 UN
s2	0 NNNN[10]	0 NNNN[10]	0 NNNN[10]
s3	0 N	0 N	0 N
s4	0 NNNNNN*	0 NNNNNN*	0 NNNNNN[60]
s5	0 NN[20]	0 NN[20]	0 NN[20]
s6	0 UUCNNNN	0 UUC****	0 NUC****
s7	0 NCCA	0 NCCA	-

TAB. 3.5 – *RnaBob.*

	d_0	d_1	d_2
	H1 s1 H2 s2 H2 s3 H3 s4 H3 s5 H4 s6 H4 H1 s7		
H1	7:7 1	7:7 2	7:7 2
H2	4:4 0	3:4 1	3:4 1
H3	5:5 0	5:5 1	5:5 1
H4	5:5 0	5:5 1	5:5 1
s1	2:2 UN	2:2 UN	2:2 UN
s2	4:14	4:14	4:14
s3	1:1	1:1	1:1
s4	6:7	6:7	6:60
s5	2:22	2:22	2:22
s6	7:7 UUC	3:7 UUC	3:7 NUC
s7	4:4 NCCA	4:4 NCCA	-

TAB. 3.6 – *RnaMot.*

d_0	d_1	d_2
h5(len=7, mispair=1) ss(len=2, seq="UN") h5(len=4) ss(minlen=4,maxlen=14) h3 ss(len=1) h5(len=5, mispair=1) ss(minlen=6,maxlen=7) h3 ss(minlen=2,maxlen=22) h5(len=5, mispair=1) ss(len=7,seq="∧UUC") h3 h3 ss(len=4,seq="NAAC")	h5(len=7, mispair=2) ss(len=2, seq="UN") h5(minlen=3,maxlen=4, mispair=1) ss(minlen=4,maxlen=14) h3 ss(len=1) h5(len=5, mispair=1) ss(minlen=6,maxlen=7) h3 ss(minlen=2,maxlen=22) h5(len=5, mispair=1) ss(minlen=3,maxlen=7,seq="∧NUC") h3 h3 ss(len=4,seq="NAAC")	h5(len=7, mispair=2) ss(len=2, seq="UN") h5(minlen=3,maxlen=4,mispair=1) ss(minlen=4,maxlen=14) h3 ss(len=1) h5(len=5, mispair=1) ss(minlen=6,maxlen=66) h3 ss(minlen=2,maxlen=22) h5(len=5, mispair=1) ss(minlen=3,maxlen=7,seq="∧NUC") h3 h3

TAB. 3.7 – *RnaMotif.*

ARNt-typeI :
902 séquences dont l'alignement a une longueur 80 bases. Il est utilisé pour les eucaryotes, archae-bactéries et bactéries.
ARNt-typeII :
207 séquences de longueur 99 bp. Les séquences composant cet alignement possède une hélice supplémentaire dans la boucle de longueur variable, mais aucune ne contient d'introns.
ARNt-allnuclear :
Ce modèle regroupe les 1110 séquences des deux modèles précédents et l'alignement a une de longueur 99 bases. Il est annoté comme étant général pour les eucaryotes, archae-bactéries et bactéries et relativement spécifique. Aucune ne contient d'introns

TAB. 3.8 – *Les trois lots de séquences d'apprentissage d'ARNt fournis avec le programme Erpin.*

```
%program {
# helix sections
helix { #Hélice A
( ge loop 44 )
( strstr ( sstr ( seq fullseq ) ( add ( end head ) 1 ) 1 ) "T" )
( strstr ( sstr ( seq fullseq ) ( add ( end tail ) 2 ) 3 ) "CCA" )}
helix { #Hélice D
( ge head 3 )
( le head 4 ) }
helix {#Hélice C
( eq head 5 )
( ge loop 6 ) }
helix {#Hélice T
( eq head 5 )
( ge loop 3 )
# La boucle de l'hélice T commence avec "TTC"
( strstr ( sstr ( seq fullseq ) ( add ( end head ) 1 ) 3 ) "TTC" ) }
# Span section
span { %end_head 3 %end_tail 2 %end_tail 23 }
# Cross section
cross {
# Le premier bras de l'hélice D commence 2 nt après le premier bras de l'hélice A
( eq ( sub ( start head #2 ) ( end head #1 ) ) 3 )
# le premier bras de l'hélice C est à 1 nt de la fin du second bras de l'hélice D
( eq ( sub ( start head #3 ) ( end tail #2 ) ) 2 )
# Le premier bras de l'hélice T est à plus de 2 nt de la fin du second bras de l'hélice C
( ge ( sub ( start head #4 ) ( end tail #3 ) ) 3 )
# Le second bras de l'hélice A commence juste après la fin du second bras de l'hélice T
( eq ( sub ( start tail #1 ) ( end tail #4 ) ) 1 ) }}
```

TAB. 3.9 – *Le descripteur d_1 selon Palingol.*

3.4.4 Résultats obtenus avec *Escherichia coli*

3.4.4.1 Données de Référence

Les premiers jeux de données utilisés sont issus de *Escherichia coli*, dont le génome a été séquencé en 1997 [BPB+97], avec une taille de 4,6 méga bases.

Parmi les 88 ARNt annotés qu'il contient:

– 86 présentent une structure canonique ;

– 1 est un pseudo-gène ;

– 1 est un ARNt de la sélénocystéine.

Nous avons utilisé parallèlement une base de données regroupant les 88 ARNt et le génome entier.

3.4.4.2 Analyse des résultats obtenus avec les logiciels généralistes

Le tableau 3.10 présente les résultats obtenus pour les 88 ARNt de *E.coli* et pour le génome de *E. coli* en utilisant les logiciels généralistes et les descripteurs d_0 et d_1.

Les nombres de vrais positifs, faux négatifs et faux positifs sont mentionnés pour nous permettre d'analyser les différences potentielles entre les différentes méthodes.

Les solutions chevauchantes

Un point important à considérer avant d'utiliser un des logiciels généralistes concerne la façon dont ils procèdent avec les solutions chevauchantes. La figure 3.3 montre 9 combinaisons de solutions pour un ARN donné localisé aux positions (a, b) sur la séquence.

Lorsque l'option chevauchement est désactivée avec les logiciels Patscan et RnaBob, l'ARN n'est pas trouvé si un candidat est trouvé aux positions représentées par les cas I, VII, VIII et IX (si un autre candidat recouvre partiellement ou entièrement l'ARN sur sa région 5'). Dans les autres cas, les solutions correspondant ou incluant l'ARN sont trouvées. En effet, ces solutions ont des positions 5' commençant après la position 5' de l'ARN.

Si les deux logiciels autorisent les chevauchements, ils adoptent une démarche différente. Patscan trouve tous les cas (à l'instar des autres logiciels ne permettant pas de paramétrer une recherche de solutions non chevauchantes). RnaBob, quant à lui, ne trouve pas l'ARN dans les cas I et II.

RnaMot propose un fichier de solution "alternatives" pour différencier les solutions correspondant aux cas II (mêmes positions de début et fin mais avec une structure différente). Il est également intéressant de noter que le programme RMPRUNE (fourni en annexe de RnaMotif) élimine les solutions chevauchantes d'une façon différente de RnaMot ou RnaBob. En effet, un candidat C_1 est considéré comme similaire à un candidat C_2 si l'ensemble des appariements contenus dans les hélices de C_1 sont des sous ensembles des appariements contenus dans les hélices de C_2.

Ainsi, le nombre de candidats peut différer entre les différents logiciels selon les différentes définitions de solutions chevauchantes ou similaires implémentées. Nous comptabilisons comme vrais positifs les candidats présentant des positions identiques avec l'ARN.

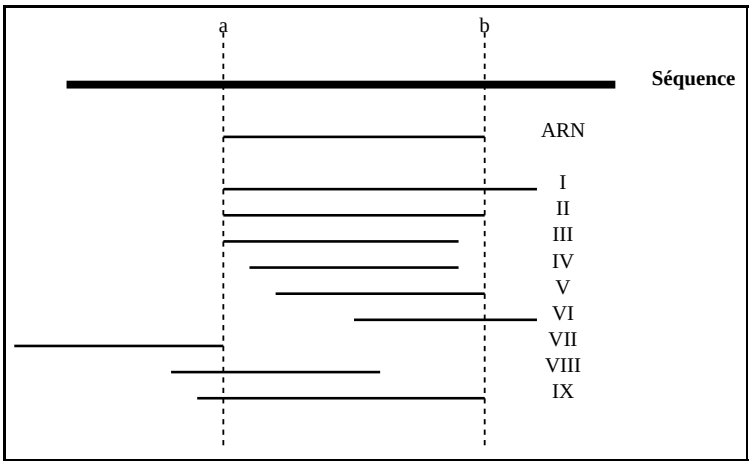


FIG. 3.3 – *Candidats chevauchant une séquence ARN localisée aux positions a et b.*

Nous distinguons parmi l'ensemble des solutions, les solutions dites "uniques" des solutions dites "doubles" (positions de début et de fin de la prédiction communes à au moins deux candidats, par exemple l'ARN et le cas II de la figure 3.3).

Le descripteur d_0

Les résultats obtenus avec le premier descripteur d_0 sont similaires pour l'ensemble des logiciels généralistes (excepté palingol) et pour les deux types de données. Les 57 solutions obtenues correspondent à de vrais positifs et aucun faux positif n'est trouvé. Ce premier descripteur est relativement peu souple, et manque 31 solutions (faux négatifs).

Palingol trouve 53 solutions. 4 solutions parmi les 57 précédemment citées sont manquées. Palamou, le programme utilisé par Palingol pour générer les hélices *a priori*, est responsable de cette différence de résultats en n'autorisant pas les mésappariements aux extrémités des hélices. Les quatre ARN de transfert ainsi manqués présentent cette caractéristique au niveau du "bras accepteur" de la structure secondaire canonique.

Le langage de RnaMotif autorise la spécification de la localisation des mésappariements, et lorsque le paramètre par défaut est utilisé (n'autorise pas les mésappariements aux extrémités), le nombre de solutions est également égal à 53.

Biologiquement, une hélice de 7 paires de base avec un mésappariement aux extrémités n'existe pas. Le fait que la plupart des logiciels généralistes prennent cette possibilité en compte permet de simplifier la description de la molécule. En effet, certains ARNt (notamment chez *E. coli* présentent une hélice A de 6 paires de bases et la région 5' de cette dernière est distante de deux bases (au lieu d'une seule base) du mot CCA. Dans ce cas de figure, les logiciels, prenant en compte les mésappariements aux extrémités, vont trouver les séquences correspondantes. Une description plus rigoureuse des ARN de transfert dans un tel cas nécessiterait de décrire:

– une hélice de 6 paires de bases distante de 2 bases du mot CCA;

– ou une hélice de 7 paires de bases distante de 1 base du mot CCA.

A l'exception de Patscan, RnaMotif et Palingol, les logiciels ne permettent pas de poser une contrainte binaire reliant la taille de l'hélice à la taille de la séquence séparant l'hélice du mot. Dans le cas de Palingol, il est ainsi possible de décrire rigoureusement les mêmes informations

contenues dans d_0 mais en rendant variables les longueurs des quatres hélices et des régions adjacentes à chaque extrémité des hélices. Le programme obtenu est très complexe.

Le descripteur d_1

Le deuxième descripteur, d_1, plus souple regroupe les différents logiciels généralistes dans 3 catégories lors d'une recherche dans la base de données et 4 catégories avec le génome.

Les résultats obtenus avec la base de données

RnaBob avec/sans chevauchement se comporte comme PatScan sans chevauchement. 87 candidats sont obtenus et ils correspondent tous à des vrais positifs. Le faux négatif est celui de la sélénocystéine décrit précédemment. Le résultat est attendu du fait de la structure non canonique de cet ARNt.

PatScan avec chevauchement, RnaMotif et RnaMot composent la deuxième catégorie comprenant 237 solutions. Le nombre de solutions uniques est de 87, identique à ceux trouvés par la première catégorie. Les 150 doubles prédictions possèdent des bornes identiques à des solutions déjà trouvées mais proposent une localisation différente des éléments composant le motif structuré.

RnaMotif (en n'autorisant pas les mésappariements aux extrémités) et Palingol trouvent des résultats identiques. Ils manquent 6 vrais positifs correspondant à l'ARNt de la sélénocystéine et à 5 ARNt possédant un mésappariement à l'extrémité d'une hélice.

Les résultats obtenus avec le génome

Ils se classent dans quatre catégories. La première comprend les résultats de PatScan et RnaBob sans chevauchement avec 215 solutions, parmi lesquels se trouvent 85 vrais positifs. Aucune solution double n'est trouvée, résultat en accord avec le caractère non exhaustif de la recherche. Trois vrais positifs sont manqués, l'un d'entre eux est l'ARNt de la sélénocystéine, et les deux autres faux négatifs ont été trouvés avec le descripteur d_0 lors d'une recherche sur la base de données. Ce résultat est lié au caractère non exhaustif de la recherche : deux solutions chevauchantes avec ces deux vrais positifs sont trouvées par les programmes, et étant trouvées avec des positions initiales en amont des vrais positifs, elles masquent ainsi leur recherche.

En autorisant les chevauchements avec RnaBob, 238 solutions sont trouvées (aucune double solution) parmi lesquelles on retrouve 87 vrais positifs.

PatScan, quant à lui, en mode de recherche exhaustif, propose 545 solutions, à l'instar de RnaMot et RnaMotif. Parmi cet ensemble de solutions, 238 sont des solutions uniques et correspondent à la totalité des solutions trouvées par RnaBob en recherche exhaustive. En effet, RnaBob autorise les chevauchements mais à l'inverse des trois autres logiciels, n'accepte pas plusieurs solutions avec une position initiale identique, au nombre de 307.

D'après ces premiers résultats, on observe un comportement différent en terme d'exhaustivité suivant le logiciel utilisé. PatScan et RnaBob paramètrent le caractère exhaustif de la recherche, mais ne l'interprètent pas de la même manière (voir page 95). PatScan et RnaBob, si le paramètre chevauchement est désactivé, ne réamorce la recherche qu'à partir de la base suivant la dernière base du dernier candidat. Si l'on autorise les chevauchements, PatScan se

comporte à l'instar des autres logiciels. Par contre, RnaBob ne considère pas les solutions avec une position initiale identique.

Les résultats de Palingol présentés pour le génome ont été effectués avec un descripteur d_1 légèrement modifié. En effet, Palamou, en recherchant *a priori* l'ensemble des hélices possibles, génère des fichiers de taille importante (532 Méga octets avec le descripteur d_1) et la recherche effectuée par Palingol devient par conséquent très peu efficace. Pour palier à ce problème, nous avons utilisé une version de Palamou intégrant des informations similaires à celles utilisées dans la section SPAN, afin de réduire le nombre d'hélices potentielles. De plus, nous avons utilisé la matrice de score d'appariements proposées par palamou. Elle permet de calculer un score d'appariements pour chaque hélice, et de filtrer les candidats ne dépassant pas la valeur seuil de score donnée pour chaque hélice. Le calcul et l'utilisation de ce score permettent de prendre en compte des informations d'appariements compensatoires et de rejeter les hélices candidates possédant simultanément des mésappariements et un certain nombre d'appariements de type Wobble.

L'utilisation d'une fonction de score et le classement des solutions

Le langage de RnaMot autorise l'utilisateur à contrôler le nombre global de mésappariements et d'appariements de type Wobble dans l'ensemble des hélices (cf tableau M=2 : nombre global de mésappariements $\leq$ 2 et W=3 : nombre global d'appariements de type Wobble $\leq$ 3). Les tests utilisant ce type de contraintes ont donné des résultats sur le génome de *E. coli* en améliorant de façon importante la spécificité (le nombre de faux positifs diminue de façon conséquente). Un autre intérêt de RnaMot, en terme d'utilisation, est de classer systématiquement les solutions dans deux fichiers, le premier pour les "solutions" et le second pour les "alternatives". Un candidat est classé solution si il est un candidat "unique" (positions uniques) ou si il a été prédit avec le meilleur score du groupe de candidats prédits avec des positions identiques. Les candidats regroupés dans les alternatives sont les autres séquences du groupe aux bornes identiques.

On obtient, en fixant le nombre maximum de mésappariements à 2 et d'appariements de type Wobble à 3, 226 solutions parmi lesquelles 87 sont "uniques" (les vrais positifs habituels) et 139 sont des doubles prédictions. Le classement effectué par RnaMot permet de distinguer ces deux groupes de solutions. Grâce à cette particularité du logiciel, nous avons pu mettre facilement en relief les vrais positifs du génome de *E. coli*.

RnaMotif permet également à l'utilisateur de définir sa propre fonction de score. Afin de comprendre les fonctions de score de RnaMot et RnaMotif, nous avons ajouté au descripteur 2 de RnaMotif une fonction pour rejeter les candidats avec un nombre global de mésappariements supérieur à 2. N'ayant pas réussi à contrôler le nombre global d'appariements de type Wobble avec RnaMotif, nous avons recommencé la recherche précédente de RnaMot avec d_2 en contrôlant seulement le nombre global de mésappariements. Les résultats obtenus avec les deux méthodes sont identiques, ils ont rejeté les mêmes solutions (le nombre de faux positifs est passé de 458 à 151). Pour classer les solutions de RnaMotif, les auteurs fournissent une application en annexe nommée RmPrune. Ce programme a pour rôle de filtrer les prédictions en éliminant les solutions redondantes définies comme ayant des hélices positionnées à l'‘intérieur d' hélices d'autres prédictions. Pour comparer leurs façons respectives de classer les solutions, nous avons utilisé rmprune. Ce dernier a rejeté 100 solutions chevauchantes à d'autres. Il a re-

tenu 138 candidats potentiels, alors que RnaMot en a retenu 89 dans son fichier principal et 149 dans le secondaire.

Malgré l'utilisation de RmPrune, le nombre de doubles prédictions trouvées par RnaMotif est toujours élevé comparé à celui obtenu avec RnaMot. Les doubles prédictions, non éliminées par le filtre de la redondance, correspondent à des candidats avec des longueurs d'hélices identiques, mais des régions simple brin différentes. Les bornes de ces prédictions étant identiques pour être considérées comme des doubles prédictions, la différence entre deux régions simple brin de deux candidats est obligatoirement compensée par une autre région simple brin de taille différente.

Erpin

Nous avons utilisé les trois lots de séquences d'apprentissage d'ARNt fournis avec le programme, ainsi qu'un jeu de données construit à partir des 57 ARN de transfert trouvés avec le descripteur 1 des logiciels généralistes. A partir de l'alignement de structure primaire et secondaire de ces 57 séquences, Erpin prédit 61 candidats, tous étant des vrais positifs. Après comparaison des séquences obtenues avec les 57 originels, nous avons remarqué qu'une des 57 séquences n'est pas retrouvée par Erpin. Ce faux négatif contient une brèche dans un élément simple brin, cas de figure unique dans l'alignement. 4 nouvelles séquences sont trouvées par Erpin (ne font pas partie des séquences fournies pour la phase d'apprentissage). Elles ont en commun la présence de mésappariements aux extrémités.

La somme du nombre de solutions obtenues avec les séquences d'apprentissage de *type I et II* est identique au nombre de solutions trouvées avec *type ARNt-allnuclear*. En effet, les modèles sont complémentaires.

Les deux ARN de transfert non trouvés par les trois types d'alignement de séquences sont celui de la sélénocystéine et le pseudo-ARN de transfert. Le nombre de faux positifs est nul, quelque soit le type d'apprentissage utilisé.

	Solutions *unique*	*double*	**Vrai Positifs**	**Faux Négatifs**	**Faux Positifs**	**Temps**
(A) : Base de données Descripteur d_0						
PatScan-RnaBob avec/sans chev.	57					<1 s.
RnaMot-RnaMotif	*57*	*0*	57	35	0	<1 s.
RnaMotif sans mésap. aux extr.	53					<1 s.
Palingol	*53*	*0*	53	35	0	<1 s.
Descripteur d_1						
RnaBob avec/sans chev.	87					<1 s.
PatScan sans chev.	*87*	*0*	87	1	0	<1 s.
PatScan avec chev.	237					<1 s.
RnaMot - RnaMotif	*87*	*150*	87	1	150	<1 s.
RnaMotif sans mésap. aux extr.	166					<1 s.
Palingol	*82*	*84*	82	6	84	21 s.
Erpin						
57 séquences	*61*	*0*	61	17	0	<1 s.
ARNt-typeI	*70*	*0*	70	18	0	<1 s.
ARNt-typeII	*16*	*0*	16	72	0	<1 s.
ARNt-allnuclear	*86*	*0*	86	2	0	<1 s.
(B) : Génome Descripteur d_0						
PatScan/RnaBob avec/sans chev.	57					13/23 s. - 9/15 s.
RnaMot/RnaMotif	*57*	*0*	57	31	0	s. - 2 min.
RnaMotif	53					
sans mésap. aux extr.	*53*	*0*	53	31	0	1 min.
Descripteur d_1						
RnaBob/PatScan sans chev.	215					
	215	*0*	85	3	130	55/56 s.
RnaBob avec chev.	238					
	238	*0*	87	1	151	1 min. 11
PatScan avec chev.	545					1 min. 32
RnaMot/RnaMotif	*238*	*307*	87	1	458	4 s. et 2 min.
RnaMot M=2 W=3	226					
	87	*139*	87	1	139	5 s.
RnaMot M=2	238					5 s.
RnaMotif M=2	*89*	*149*	87	1	151	6 min.
Palingol	94					
descripteur d_1'	*82*	*12*	82	6	12	5 heures
Erpin						
57 séquences	*61*	*0*	61	17	0	14 s.
ARNt-typeI	*70*	*0*	70	18	0	33 s.
ARNt-typeII	*16*	*0*	16	72	0	18 s.
ARNt-allnuclear	*86*	*2*	86	2	2	1 min. 59

TAB. 3.10 – *Résultats obtenus avec les logiciels généralistes en recherchant les ARN de transfert (A) : parmi la base de données des 88 ARNt de Escherichia coli et (B) : dans le génome de Escherichia coli.* **Légende :** *(chev.) : chevauchement, (mésap.) : mésappariement, (sans mésap. aux extr.) : les mésappariements ne sont pas autorisés aux extrémités des hélices, (M=2) : nombre global de mésappariements inférieur ou égal à 2, (W=3) : nombre global d'appariements de type Wobble inférieur ou égal à 3.*

3.4.4.3 Analyse des résultats obtenus avec les logiciels spécifiques

tRNAscan-SE

Dans la partie du tableau correspondante, les résultats obtenus avec tRNAscan-SE et les logiciels qu'il utilise sont regroupés dans quatre catégories. Dans les deux premiers groupes, nous avons testé le paramètre responsable du choix de modèle de covariance (propre à un type d'organisme) en utilisant les programmes de pré-filtre (tRNAscan et EutfindtRNA génèrent une première liste de candidats sur lesquels est appliqué le modèle de covariance, Cove), puis en utilisant seulement Cove.

Les deux groupes suivants correspondent à l'utilisation spécifique de EutfindtRNA et tRNAscan. Nous avons utilisé ces deux programmes de deux façons : (i) au travers de tRNAscan-SE en utilisant les options prévues à cet effet, (ii) en utilisation directe (résultats en italique).

Les résultats donnés avec tRNAscan-SE (utilisation des trois programmes) et Cove sont identiques pour les deux jeux de données, le nombre de faux positifs est nul et seul le temps d'exécution varie suivant si l'on s'intéresse à la base de données ou au génome.

En utilisant tRNAscan-SE avec les trois logiciels intrinsèques, on obtient les 88 ARNt avec le modèle de covariance bactérien et organelle. Le temps d'exécution entre les deux varie considérablement du fait du mode organelle qui n'utilise pas de pré-filtre et exécute directement cove. Le pseudo ARNt est manqué par les trois autres modes et celui de la sélénocystéine par le mode eucaryote et général.

tRNAscan-SE utilise en effet deux procédures parallèles pour rechercher les ARNt de sélénocystéine. La première est intrinsèque à EutfindtRNA et la seconde à Cove. Le mode organelle n'utilise pas de pré-filtre et effectue sa recherche avec le modèle de covariance général et un score particulier. Les résultats suivants ont été obtenus explicitement avec cove et différents modèles de covariance. l'ARNt est trouvé alors par le modèle général et est manqué avec les autres modèles (cas identique avec le mode organelle) y compris celui des bactéries.

Nous avons ensuite utilisé les paramètres permettant de sélectionner plus spécifiquement l'un des deux filtres, tRNAscan ou EutfindtRNA, et nous avons comparé les solutions obtenues à celles générées par l'utilisation directe de ces mêmes programmes.

EutfindtRNA

On observe une faible différence avec la base de données entre les résultats obtenus avec le mode relâché ou strict (respectivement, les nombres de vrais positifs sont 81 et 82). La différence des résultats avec le génome pour ces deux modes est plus importante. 44 vrais positifs sont trouvés avec le mode strict et 82 avec le mode relâché. Ce résultat est lié à la définition du mode "strict" de EutfindtRNA qui prend en compte la recherche du site potentiel de terminaison de la polymerase III (en aval des ARNt), et dans le cas des bactéries, l'ARN polymerase III n'existe pas (une seule polymérase intervient, Page 6) . Le mode strict de EutfindtRNA ne peut pas fonctionner sur les séquences de la base de données, les séquences étant tronquées (les sites ne sont pas recherchés), et ne semble pas pénaliser les candidats trouvés. Par contre, sur le génome, on observe une diminution importante du nombre de vrais positifs lorsque le programme recherche ces sites, et le nombre de solutions est beaucoup plus faible. Ces sites sont spécifiques des eucaryotes et avec une bactérie, il est préférable d'utiliser le mode relâché.

L'utilisation directe de EutfindtRNA avec la base de données donne quelques différences, notamment, le nombre de faux positifs augmente. On peut penser que tRNAscan-SE filtre les résultats obtenus par EutfindtRNA avec un score, avant de les proposer à Cove.

tRNAscan

Les résultats obtenus par tRNAscan sont meilleurs avec la base de données. Dans le meilleur des cas, il ne trouve pas l'ARNt de la sélénocystéine et le pseudo-ARNt. L'utilisation directe ou non du programme ne donne pas beaucoup de différences, excepté pour le nombre de faux positifs. Dans ce cas aussi, on peut penser que tRNAscan-SE filtre les résultats de tRNAscan.

Par contre, lors de la recherche sur le génome, les deux modes d'utilisation (direct et indirect) du programme trouvent des résultats très différents. L'utilisation indirecte de tRNAscan avec le mode relâché génère un nombre de vrais positifs beaucoup plus faible (26 au lieu de 86). En observant en détail les résultats, nous avons remarqué que certains ARNt sont contenus dans des candidats et non comptabilisés comme de vrais positifs. En effet, ils ont une position initiale identique mais les séquences proposées sont beaucoup plus longues. Nous n'avons pas trouvé dans la littérature la description des modes strict et relâché. Une hypothèse, pour expliquer cette différence de comportement avec le même mode, serait de penser que le mode relâché lors d'une utilisation directe génère des candidats beaucoup plus longs, avec la garantie que Cove saura, le cas échéant, trouver la structure exacte de la séquence en essayant de la calquer sur son modèle.

FAStRNA

Les deux versions de FAStRNA donnent le même nombre de vrais positifs à un candidat près. La version utilisant des matrices de probabilités trouve un candidat de plus. Les trois ARNt manquants sont celui de la thréonine, le pseudo et la sélénocystéine. Le nombre de faux positifs est très faible, et comprend principalement des doubles prédictions.

Aragorn

Ce programme propose une très bonne spécificité et il manque seulement deux ARNt. Le premier correspond à celui de la thréonine, le seul ARNt avec 2 mésappariements dans l'hélice de l'amino-acyl et le second correspond au pseudo ARNt.

	Solutions		**Vrai**	**Faux**	**Faux**	**Temps**
	unique	*double*	**Positifs**	**Négatifs**	**Positifs**	
(A) : Base de données						
tRNAscan-SE						
Bactérie	88	0	88	0	0	44 s.
Archae bactérie	87	0	87	1	0	48 s.
Eucaryotes	86	0	86	2	0	40 s.
Général	86	0	86	2	0	37 s.
Organelles	88	0	88	0	0	59 s.
Cove						
Bactérie	87	0	87	1	0	64 s.
Archae bactérie	86	0	86	2	0	67 s.
Eucaryotes	86	0	86	2	0	63 s.
Général	88	0	88	0	0	61 s.
tRNAscan	84	0	84	4	0	s.
tRNAscan relâché	86	0	86	2	0	s.
EutfindtRNA	81	0	81	7	0	s.
EutfindtRNA relâché	82	0	82	6	0	s.
tRNAscan	85	1	84	4	1	s.
tRNAscan relâché	92	6	86	2	6	s.
EutfindtRNA strict/rel.	81	0	81	7	0	s.
FAStRNA-CM	92					
	85	7	85	3	7	s.
FAStRNA-CLASS	88					
	84	4	84	4	4	s.
Aragorn	86	0	86	2	0	s.
(B) : Génome						
tRNAscan-SE						
Bactérie	88	0	88	0	0	4 min. 16
Archae bactérie	87	0	87	1	0	4 min. 53
Eucaryotes / Général	86	0	86	2	0	2 min.
Organelles	88	0	88	0	0	47 h.
Cove						
Bactérie / Général	88	0	88	0	0	49 h - 47 h.
Archae / Euc.	86	0	86	2	0	49 h - 46 h.
tRNAscan	86	0	84	4	2	35 s.
tRNAscan relâché	6068	0	26	62	6042	18 min.
EutfindtRNA	47	0	44	44	3	s.
EutfindtRNA relâché	100	0	82	6	18	s.
tRNAscan	87	0	84	4	3	17 s.
tRNAscan relâché	8381	21228	86	2	29523	2 min. 12
EutfindtRNA	69	0	67	21	2	5 s.
EutfindtRNA relâché	90	0	81	7	9	8 s.
FAStRNA-CM	94					
	87	7	85	3	9	6 s.
FAStRNA-CLASS	90					
	86	4	84	4	6	6 s.
Aragorn	86	0	86	2	0	5 s.

TAB. 3.11 – *Résultats obtenus avec les logiciels spécifiques en recherchant les ARN de transfert (A) : parmi la base de données des 88 ARNt de Escherichia coli et (B) : dans le génome de Escherichia coli.* **Légende :** *tRNAscan et EutfindtRNA ont été utilisés spécifiquement par le biais de tRNAscan-SE, ou en exécution autonome (résultats en italique).*

3.4.5 Résultats obtenus avec *Saccharomyces cerevisiae*

3.4.5.1 Données de référence

Nos tests ont ensuite été effectués à partir de données issues de *Saccharomyces cerevisiae*, séquencé en 1996. Le génome de cet eucaryote est composé de 16 chromosomes, avec une taille de 12 méga bases.

275 gènes non codant sont annotés comme étant des ARN de transfert :

- 273 présentent une structure canonique ;
- 2 ont un isotype indéterminé.

Parmi les 273 ARNt, 59 possèdent un intron situé dans la boucle de l'hélice de l'anti-codon.

Ces nouveaux tests nous apportent de nouvelles informations intéressantes au regard de la taille du génome considéré et de la dégénérescence que nous avons dû ajouter aux descripteurs pour permettre la prise en compte d'un intron potentiel.

Le nombre de solutions trouvées lors de la recherche sur le génome est très impressionnant pour les logiciels généralistes. Notre objectif n'a pas été dans ce cas de tendre vers une meilleure sensibilité et spécificité mais d'essayer de comprendre les différences d'exhaustivité des logiciels généralistes.

3.4.5.2 Analyse des résultats obtenus avec les logiciels généralistes

Nous avons utilisé le descripteur d_2 pour rechercher les ARNt dans la base de données et le génome de la levure. Ce descripteur, à la différence des deux précédents, autorise l'insertion d'un intron dans la boucle de l'hélice de l'anti-codon.

Les résultats obtenus avec la base de données

La base des ARNt de la levure avec 275 séquences est trois fois plus grande que celle de *E.coli* et laisse présager des résultats avec un nombre plus important de faux positifs. En effet, la recherche avec PatScan/RnaMot/RnaMotif génère 150 doubles prédictions à partir des 88 séquences de *E.coli* et 2051 pour les 275 séquences de la levure. L'augmentation de la taille du jeu de données provoque un accroissement important du nombre de solutions. Le nombre de catégories dans lesquelles nous avons classé les résultats est de ce fait plus important et met en relief de nouvelles disparités de fonctionnement entre les différents logiciels.

Les résultats de RnaMotif sont du même ordre de grandeur que ceux obtenus avec RnaMot et PatScan, mais présentent une différence de 5 solutions. RnaMotif ne permet pas de paramétrer spécifiquement la recherche de solutions sur le brin direct des séquences d'un jeu de données. Il effectue dans tous les cas une recherche dans les deux sens. Si on paramètre RnaMot et PatScan pour rechercher sur les deux brins des séquences, on obtient alors les mêmes résultats.

PatScan, RnaMot et RnaMotif trouvent 274 vrais positifs et manquent un seul candidat correspondant à l'ARN de transfert 13 du chromosome 14 de la levure, et ayant comme isotope l'asparagine. Cet ARN de transfert a la particularité de contenir trois mésappariements dans l'hélice H1, et ce cas de figure n'est pas décrit par le descripteur 3 (le maximum est de 2).

Le nombre de faux négatifs obtenus avec Palingol et RnaMotif (sans mésappariement aux extrémités) est plus important comparé aux autres logiciels généralistes, et la raison est iden-

tique à celle invoquée pour les résultats obtenus avec le génome de *E.coli*. Les résultats des deux logiciels sont dans le même intervalle de grandeur, mais présentent une petite différence.

Un dernier point concerne le temps d'exécution de Palingol, largement supérieur aux autres temps, toutes méthodes confondues. La génération *a priori* de toutes les hélices est un handicap dont l'importance s'accroît avec la taille des données.

Les résultats obtenus avec le génome

En effectuant nos tests sur le génome entier de la levure, nous avons rencontré un problème d'espace mémoire lié à la taille des fichiers de solutions générées et les temps de calcul nécessaires sont plus importants. Pour ces raisons, nous n'avons pas été en mesure d'utiliser Palingol.

Sur le génome, la recherche est effectuée sur les deux sens de transcription par tous les logiciels. De ce fait, les résultats de RnaMotif sont identiques à ceux de PatScan. RnaMot, quant à lui, trouve trois solutions supplémentaires.

Le nombre de solutions très élevé est devenu prohibitif et met en évidence l'importance d'une fonction de score pour rejeter certaines solutions. En effet, le simple contrôle du nombre global de mésappariements et d'appariements de type Wobble permet de diminuer le nombre de faux positifs d'un facteur de 400 fois moins. De plus, l'élimination de la redondance par le classement des solutions selon RnaMot nous permet de descendre le nombre de solutions à 7537 (une différence de trois avec la valeur des solutions uniques dans le tableau, dû a trois solutions "uniques" avec une seule borne différente). Le nombre total de solutions est toujours prohibitif. Cependant il est intéressant d'observer une amélioration aussi rapide.

Afin de comparer RnaMot et RnaMotif, nous avons réédité la recherche en spécifiant le nombre maximum de mésappariements à autoriser. Les résultats obtenus avec les deux méthodes sont similaires. Il est cependant intéressant de remarquer l'efficacité croissante de RnaMot en terme de vitesse d'exécution lorsque l'on rajoute des contraintes globales (92 heures, 6h20 et 3h dans le dernier cas). RnaMot optimise sa recherche de solutions, en inspectant au fur et à mesure les contraintes globales posées sur les variables, pour élaguer l'arbre de recherche. Dans le cas de RnaMotif, la recherche sans contraintes globales se fait en 8h40, et l'utilisation d'une fonction de score a permis de diminuer le temps à 6h20. Le gain est moins important, mais le temps de recherche est globalement plus intéressant avec RnaMotif.

D'une façon générale, la recherche sur des données de taille importante met en évidence de nouvelles différences (infime, certes) entre les descripteurs "équivalents" dans les différentes syntaxes.

Base de données	**Solutions** *unique*	**Solutions** *double*	**Vrai Positifs**	**Faux Négatifs**	**Faux Positifs**	**Temps**
(A) : Base de données Descripteur d_2						
PatScan sans chev.	274					0,8 s.
RnaBob sans chev.	*274*	*0*	274	1	0	<1 s.
RnaBob avec chev.	331					
	331	*0*	274	1	57	<1 s.
PatScan avec chev.	2385					2 s.
RnaMot	*334*	*2051*	274	1	2111	2 s.
RnaMotif	2390					
	339	*2051*	274	1	2116	2 s.
Palingol	937					15 min.
RnaMotif sans mésap. aux extr.	*284*	*653*	269	6	668	2 s.
RnaMotif M=2	1734					s.
RnaMot M=2	*276*	*1458*	276	1	1460	1 s.
Erpin						
ARNt-typeI	*192*	*0*	192	83	0	1 s.
ARNt-typeII	*22*	*0*	22	253	0	4 s.
ARNt-allnuclear	*222*	*0*	222	53	0	5 s.
(B) : Génome Descripteur d_2						
PatScan sans chev.	129031					
	129031	*0*	37	238	128994	48 min.
RnaBob sans chev.	129943					
	129943	*0*	31	244	129912	24 min.
RnaBob avec chev.	804044					
	804044	*0*	257	17	803787	43 min.
PatScan avec chev.	7313791					1h40
RnaMotif	*849979*	*6463812*	274	1	7313517	8h40
RnaMot	7313806					
	849982	*6463824*	274	1	7313532	92 h.
RnaMotif M = 2	126747					6h20
RnaMot M = 2	*40846*	*85901*	274	1	126473	4h10
RnaMot	17586					
M = 2 W=3	*7540*	*9946*	274	1	17312	3 h.
Erpin						
ARNt-typeI	*192*	*0*	192	83	0	1 min.
ARNt-typeII	*22*	*0*	22	253	0	48 s.
ARNt-allnuclear	*222*	*0*	222	53	0	3 min.

TAB. 3.12 – *Résultats obtenus avec les logiciels généralistes en recherchant les ARN de transfert (A) : parmi la base de données des 275 ARNt de Saccharomyces cerevisiae et (B) : dans le génome de Saccharomyces cerevisiae.* **Légende :** *(chev.) : chevauchement, (mésap.) : mésappariement, (sans mésap. aux extr.) : les mésappariements ne sont pas autorisés aux extrémités des hélices, (M=2) : nombre global de mésappariements inférieur ou égal à 2, (W=3) : nombre global d'appariements de type Wobble inférieur ou égal à 3.*

3.4.5.3 Analyse des résultats obtenus avec les logiciels spécifiques

tRNAscan-SE

Nous avons effectué la même série de tests présentés pour *E. coli*.

– Le mode bactérien sur le génome de *E. coli* a donné 13 faux négatifs. Parmi eux l'asparagine avec 3 mésappariement dans l'hélice H1 (non trouvés par l'ensemble des méthodes généralistes) et 12 ARN de transfert avec intron. Les autres ARN de transfert avec intron sont trouvés mais l'intron n'est pas considéré comme tel (non recherché dans le cas des bactéries). Les positions des ARNt prédits diffèrent dans certains cas de trois bases avec les ARNt annotés. En effet, le mode bactérien recherche le motif CCA (où s'accroche l'acide aminé) sur le gène. Ces bases sont ajoutées la plupart du temps chez les eucaryotes lors de la maturation de l'ARNt par une protéine.
– Il est intéressant de remarquer que le mode organelle génère des faux positifs sur le génome. Ces prédictions doivent être dues au filtre utilisé pour le score (diminué pour ce mode).
– L'utilisation parallèle des programmes tRNAscan et EutfindtRNA directe ou indirecte donne des résultats similaires en terme de vrais positifs, et pour les deux jeux de données. La différence est liée seulement au nombre de faux positifs pour les raisons proposées précédemment.

  tRNAscan en mode strict ne trouve pas 5 ARNt à la structure un peu particulière. Le fait de relâcher les paramètres permet d'en trouver 2 parmi eux. Le nombre de faux positifs est important dans la seconde recherche.

  EutfindtRNA trouve (mode strict ou relâché) l'ensemble des vrais positifs avec un nombre de faux positifs beaucoup plus modéré.

Erpin

Les résultats obtenus avec Erpin sont identiques avec les deux jeux de données, et aucun faux positif n'est trouvé. Les séquences constituant les trois types d'alignement fournis à Erpin pour sa phase d'apprentissage, sont exclusivement des ARNt sans intron. On obtient 214 vrais positifs en faisant la somme des résultats obtenus avec les alignements de *ARNt-type I et -type II* et 222 vrais positifs avec l'alignement *ARNt-type allnuclear*. Parmi les 275 ARNt de la base, 216 n'ont pas d'intron parmi lesquels 214 sont trouvés avec les trois types d'alignements. Les deux séquences sans intron manquantes correspondent à deux ARNt dont l'isotype n'a pas été déterminé .

D'après ces résultats, le descripteur allnuclear nous a permis de trouver 8 ARNt avec intron (222 = 214 + 8). Après analyse de ces prédictions, il s'avère qu'Erpin prédit par le plus grand hasard ces ARNt avec intron. En effet, les alignements d'apprentissage ne contenant aucun ARNt avec intron, les ARNt avec intron prédits ne le sont pas avec la structure secondaire correcte (cf tRNAscan-SE). Le deuxième bras de l'hélice de l'anticodon n'est pas prédit à la bonne position, et la région variable entre l'hélice anticodon et Psi est plus importante dans la mauvaise prédiction.

FAStRNA

Les deux versions du programme manquent les mêmes ARNt. FAStRNA est le logiciel le plus performant sur le plan de la vitesse d'exécution.

Aragorn

Un seul ARN de transfert n'est pas trouvé par ce programme. Il correspond à celui de l'asparagine avec trois mésappariements dans l'hélice H_1. La spécificité de ce logiciel est de 100% sur la base de données et le génome de la levure.

3.4.6 Discussion sur la recherche des ARNt

3.4.6.1 Améliorer la sensibilité et la spécificité des logiciels généralistes

Notre principal objectif avec les logiciels généralistes (à l'exception de Cove et Erpin) a été de proposer des descripteurs similaires pour les langages fournis par chaque logiciel et de pointer les différences entre les résultats. La comparaison entre logiciels spécifiques et généralistes ne peut être effectuée à partir de l'étude que nous avons menée, et il est de ce point de vue normal que les spécifiques apparaissent dans l'ensemble des tests avec une meilleure spécificité.

Cependant, il est toujours possible d'améliorer la sensibilité et spécificité en relâchant ou contraignant respectivement le descripteur (selon la connaissance des caractéristiques de la famille de molécules d'intérêt).

Dans le cas de la famille d'ARNt, les ARNt manqués sont toujours :

– l'ARNt de la sélénocystéine pour *E. coli*

– l'ARNt de l'arginine pour *S. cerevisiae*

Le cas de l'ARNt de la sélénocystéine est résolu en autorisant une hélice A plus longue (7 à 8 paires de bases) et en remplaçant la base invariante U de la boucle D par une base quelconque.

A partir de la même idée, l'ARNt de l'arginine manqué pour *S. cerevisiae* est trouvé si le descripteur de l'hélice A autorise un mésappariement supplémentaire. Ces modifications sur les différents descripteurs concernés permettent ainsi d'améliorer la sensibilité de la recherche. Cependant la flexibilité ajoutée diminue la spécificité.

Une façon d'obtenir une meilleure spécificité pour la recherche d'ARNt consiste à observer que le nombre total de mésappariements autorisés (5 pour 4 hélices) est supérieur au nombre habituellement trouvé dans les ARNt (au maximum 3 mésappariements). La possibilité de jouer avec cette notion apparaît de ce fait un avantage.

Ainsi, Rnamot permet de contraindre le nombre total de mésappariements ainsi que le nombre total d'appariements de type Wobble (dont un trop grand nombre rend improbable le candidat concerné). Ainsi, en contraignant le nombre total à 2 mésappariements, le nombre de faux positifs diminue d'un facteur de 2,4 pour *E. coli* et de 415 pour *S. cerevisiae*.

La fonction de score proposé par Rnamotif ou la section CROSS de Palingol permettent d'exprimer également ces contraintes globales, ainsi que d'autres plus complexes. Palamou, le générateur d'hélices de Palingol, permet également d'attribuer un score à chaque type d'appariement et de filtrer les hélices pour un seuil donné.

3.4.6.2 Un exemple du potentiel d'un langage déclaratif

Il est intéressant de noter les capacités d'un langage déclaratif en potentiel d'expression. Palingol illustre parfaitement ce propos en offrant un certain nombre d'opérateurs et d'instructions de choix et de boucle à l'instar d'un langage de programmation. Par exemple :

- l'instruction conditionnelle "Si (condition vérifiée) Alors (exécute instruction 1) Sinon (exécute instruction 2)" sélectionne les lignes du descripteur à évaluer.
- l'instruction "Tant Que (condition vérifiée) (exécute instruction)" réitère un bloc d'expression tant qu'une condition est vraie.

Un illustration est donnés dans la figure 3.4 où deux structures de type tige-boucle sont représentées. Palingol permet de modéliser des contraintes assez complexes, comme par exemple:

$$\begin{cases} \text{longueurs des tiges } 1 \text{ et } 2 \geq 5 \\ 4 \leq \text{ longueur de la boucle } 1 \leq 6 \\ \text{longueur de la boucle } 2 \geq \text{ double de la longueur de la boucle } 1 \\ \text{longueur de l'espaceur entre les deux tiges est un multiple de } 3 \\ \text{la boucle } 2 \text{ contient le complementaire inverse de la boucle } 1 \end{cases}$$

Un autre exemple biologique est donné par [HEVM00] dans le cadre d'une recherche de triple hélice.

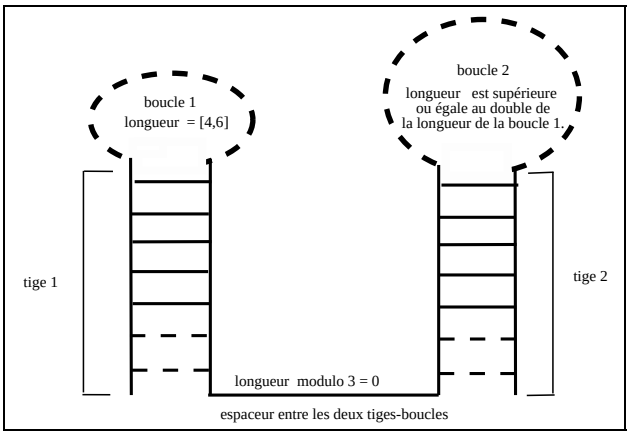


FIG. 3.4 – *Structure hypothétique mettant en relief les avantages d'un langage déclaratif.*

3.4.6.3 Conclusion

Cette première analyse s'est révélée informative sur les différents logiciels utilisés. Concernant les spécificité et sensibilité, on remarque une grande différence entre les logiciels généralistes et spécifiques. Ce résultat est biaisé du fait des informations utilisées pour décrire la molécule. Nous aurions pu essayer d'intégrer les informations utilisées par les logiciels spécifiques aux descripteurs des logiciels généralistes, mais cela n'a pas été notre propos. En effet, tous ne proposaient pas un système de score pour évaluer les candidats (un des points majeurs des logiciels spécifiques qui peuvent ainsi pondérer de façon spécifique la présence d'éléments particuliers), ajouté au fait que la tâche aurait été laborieuse. Il nous aurait été impossible de

comparer ces logiciels en utilisant des descripteurs différents. De plus, l'utilisation de ce système de score peut s'avérer assez coûteux en terme de temps pour comprendre le langage de façon précise.

Nous avons retenu comme points principaux, lors de notre analyse :

- l'exhaustivité : les comportements de la plupart des logiciels semblent similaires à quelques petites différences observées, et pas toujours explicables ;
- l'efficacité de la recherche : en terme de vitesse des parcours d'arbres de recherche des logiciels généralistes présente des différences d'un logiciel à l'autre. Il est intéressant de remarquer l'amélioration du temps d'exécution de RnaMot lorsqu'une contrainte globale est ajoutée à son descripteur ;
- la sensibilité : dans le cadre de l'analyse portant sur les logiciels spécifiques, la meilleure sensibilité est observée avec tRNAscan-SE, lequel combine plusieurs méthodes.

	Solutions *unique*	**Solutions** *double*	**Vrai Positifs**	**Faux Négatifs**	**Faux Positifs**	**Temps**
(A) : Base de données						
tRNAscan-SE						
Bactérie	262	0	262	13	0	3 min.
Archae bactérie	274	0	274	1	0	3 min. 16
Général / Eucaryotes / Organelle	275	0	275	0	0	2 min./3 min.
Cove						
Bactérie	262	0	262	13	0	3 min. 43
Archae bactérie	274	0	274	1	0	4 min. 16
Général Eucaryotes	275	0	275	0	0	3 min. 29
tRNAscan	270	0	270	5	0	s.
tRNAscan relâché	272	0	272	3	0	s.
EutfindtRNA	275	0	275	0	0	s.
EutfindtRNA relâché	275	0	275	0	0	s.
tRNAscan	270	67	270	5	67	s.
tRNAscan relâché	272	75	272	3	75	s.
EutfindtRNA	275	0	275	0	0	s.
EutfindtRNA relâché	275	29	275	0	29	s.
FAStRNA-CLASS	336					
	272	64	272	3	64	s.
FAStRNA-CLASS	310					
	272	38	272	3	38	s.
Aragorn	274	0	274	1	0	3 s.
(B) : Génome						
tRNAscan-SE						
Bactérie	262	0	262	13	0	9 min.
Général / Eucaryotes	275	0	275	0	0	5 min.
Archae Bactérie	274	0	274	1	0	10 min.
Organelles	282	0	275	0	7	5 jours
Cove						
Eucaryotes	275	0	275	0	0	5 jours
Général	279	0	275	0	4	5 jours
Bactérie	262	0	262	13	0	5 jours
Archae Bactérie	274	0	274	1	0	5 jours
tRNAscan	275	0	269	6	6	1 min.
tRNAscan relâché	15073	0	144	131	14929	18 min.
EutfindtRNA	284	0	275	0	9	21 s.
EutfindtRNA relâché	332	0	275	0	57	20 s.
tRNAscan	276	89	270	5	95	45 s.
tRNAscan relâché	21554	59254	272	3	80536	4 min.
EutfindtRNA	284	0	275	0	9	20 s.
EutfindtRNA relâché	304	0	275	0	29	18 s.
FAStRNA-CLASS	336					
	272	64	272	3	64	13 s.
FAStRNA-CLASS	310					
	272	38	272	3	238	13 s.
Aragorn	274	0	274	1	0	3 min.

TAB. 3.13 – *Résultats obtenus avec les logiciels spécifiques en recherchant les ARN de transfert (A) : parmi la base de données des 275 ARNt de S. cerevisiae et (B) : dans le génome de S. cerevisiae* . **Légende :** *tRNAscan et EutfindtRNA ont été utilisés spécifiquement par le biais de tRNAscan-SE, ou en exécution autonome (résultats en italique).*

3.5 Recherche de snoARN à boîtes C/D

La deuxième partie de cette analyse porte sur les petits ARN nucléolaires à boîtes C/D. Nous avons choisi cette molécule peu structurée pour nous intéresser à la recherche d'éléments particuliers, tels que l'interaction à distance entre une région du snoARN et de l'ARN cible.

Dans le cadre de ce travail, nous avons adopté une approche de biologiste en essayant d'identifier la meilleure méthode pour rechercher des molécules appartenant à cette famille. En effet, à notre connaissance, un seul logiciel spécifique (Snoscan) existe et il est plus particulièrement adapté à un organisme donné. Aussi, nous avons adopté parallèlement deux logiciels généralistes. Parmi les logiciels généralistes auxquels l'utilisateur à la charge de fournir un descripteur, nous avons utilisé un seul des programmes. En effet, l'étude présente n'apporte aucune information supplémentaire sur leurs différences, les caractéristiques définissant la famille des snoARN étant moins structurés. Notre choix s'est porté sur Patscan pour des raisons d'exhaustivité et d'efficacité en terme de temps. Le second logiciel généraliste utilisé est Erpin adoptant une approche probabiliste. Nous l'avons utilisé en lui fournissant pour sa phase d'apprentissage un alignement de snoARN d'archae-bactéries à boîtes C/D fournis par les auteurs [20].

Pour chacune des approches, nous avons évalué la sensibilité et spécificité obtenus avec nos jeux de données.

3.5.1 Description de la molécule

Chez les eucaryotes, durant les processus de transcription, de nombreuses modifications post-transcriptionnelles sont introduites à des positions spécifiques de certaines molécules d'ARN. Parmi ces modifications, les plus fréquentes sont de loin la 2'-O méthylation et l'isomérisation de l'uridine en pseudo-uridine. La découverte de familles de petits ARN nucléolaires (snoARN) spécialisés dans le positionnement précis de chacune de ces modifications a suscité depuis quelques années un vif intérêt quant aux mécanismes et au rôle biologique des modifications de l'ARNr.
Les petits ARN guides de modification se répartissent en deux familles selon la nature de la modification qu'ils guident et leur structure. La première famille, à laquelle nous nous intéressons ici, contient les petits ARN antisens à boîtes C/D (Fig .3.5)

Ils se caractérisent par la présence de deux motifs consensus RUGAUGA (ou boîte C) en extrémité 5' et CUGA (ou boîte D) en 3', ainsi que par une autre copie de chacun de ces 2 motifs (C' et D') dans la région centrale. Ils contiennent aussi immédiatement en amont de la boîte D (ou D') un segment de 10 à 20 nucléotides (dit éléments antisens) pouvant s'apparier à leur cible (ARNr, snARN ...) pour former un duplex. Remarquablement, la méthylation est toujours dirigée sur le nucléotide apparié à la cinquième position en amont de la boîte D dans le duplex formé avec la cible mais la base structurale de cette règle de positionnement n'est pas connue. Enfin, la plupart de ces snoARN contiennent une hélice terminale de 4 à 5 paires de bases, rapprochant les boîtes C et D pour former un motif structural caractéristique reconnu par une protéine.

20. URL : http://tagc.univ-mrs.fr/erpin/search.html

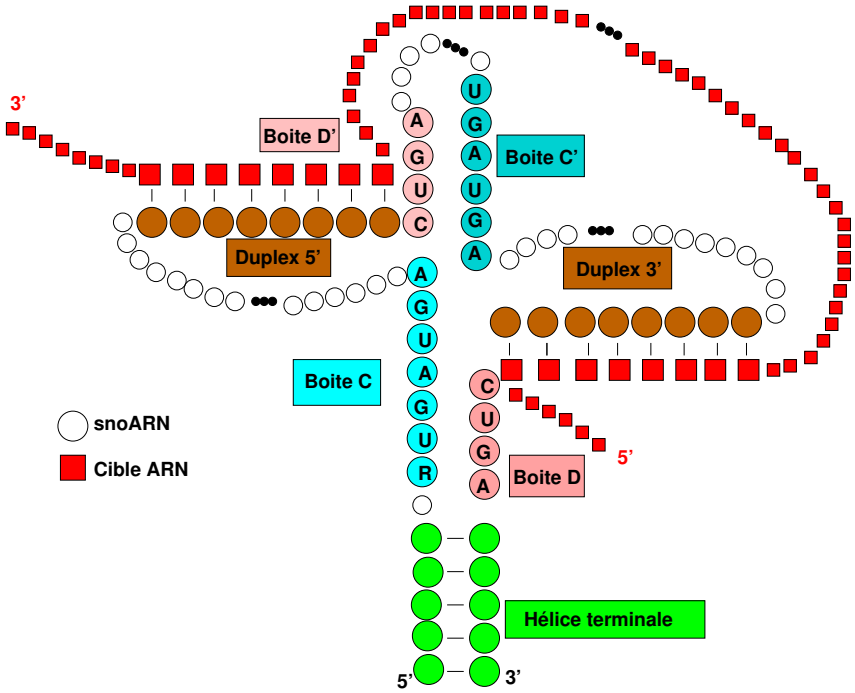


FIG. 3.5 – *Structure secondaire d'un snoARN antisens à boîtes C/D et de l'ARN cible avec lequel il forme un duplex.*

3.5.2 Un algorithme sur mesure pour les snoARN à boîtes CD

Snoscan [LE99] est un logiciel dédié à la recherche de snoARN à boîtes C/D. Il s'appuie sur un modèle probabiliste pour identifier les différents éléments des snoARN à boîtes C/D et intègre un algorithme de recherche de similarités entre les régions en amont des boîtes D et une banque d'ARN cibles, pour identifier les bases pouvant être 2'-O méthylées. L'algorithme de Snoscan effectue séquentiellement la recherche de 6 composants en utilisant des Chaînes de Markov :

- Boîte D
- Boîte C
- Duplex
- Boîte D' si le duplex n'est pas directement adjacent à la boîte D
- Position à méthyler
- Hélice terminale

Le programme prend aussi en compte la distance relative entre les caractéristiques identifiées au sein du snoARN pour diminuer le nombre de faux positifs.

3.5.3 Les modèles et protocoles de recherche utilisés par les logiciels généralistes

Dans le cadre de cette étude, nous avons utilisé essentiellement deux logiciels généralistes : PatScan et Erpin. Les outils généralistes ne permettent de spécifier ni des relations intermoléculaires ni la présence d'éléments optionnels. De ce fait, ni l'existence du duplex formé entre le petit ARN et sa cible ni la présence optionnelle d'une hélice terminale ne peuvent être utilisées. De plus, la présence des boîtes C' et D' peut être optionnelle, certains snoARN n'en étant pas pourvus ou bien elles sont trop dégénérées pour être identifiées.

Descripteurs strict et relâché de PatScan

Descripteur strict : Dans le cas des deux archae-bactéries étudiées (*Pyrococcus abyssi* et *Pyrobaculum aerophilum*), nous avons construit une expression régulière (Fig 3.6) à partir de l'alignement multiple des 59 snoARN de *Pyrococcus abyssi*. Pour *Saccharomyces cerevisiae* (Fig 3.7) nous avons utilisé un descripteur spécifiant les deux séquences consensus des boîtes C et D en autorisant une erreur pour chaque boîte.

Descripteur relâché : Par cette appellation, nous entendons une double utilisation de PatScan avec deux types de descripteurs.

Le premier contient la spécification des deux boîtes C et D, séparées par une certaine distance et en autorisant une erreur sur chacune d'entre elles. A partir de chaque solution obtenue lors de l'utilisation de ce premier descripteur, nous avons construit une base de données *ad-hoc* en ajoutant en fin de chaque candidat les trois ARN ribosomiques. La banque de données ainsi constituée contient le nombre de candidat initial multiplié par trois.

Le second descripteur (Fig 3.8 pour les archae-bactéries et pour la levure Fig 3.9) utilisé par Patscan sur cette base de données *ad-hoc*, prend en compte le duplex pouvant se former entre le candidat et l'ARN cible en recherchant une hélice dont le premier bras doit se situer en amont de la boîte D et le second n'importe où après la région correspondant au snoARN (correspond à la deuxième partie de la séquence constituée par l'ARN cible).

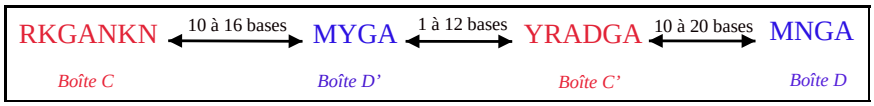


FIG. 3.6 – *Descripteur strict de PatScan pour rechercher les snoARN à boîtes C/D dans Pyrococcus abyssi et Pyrobaculum aerophilum.*

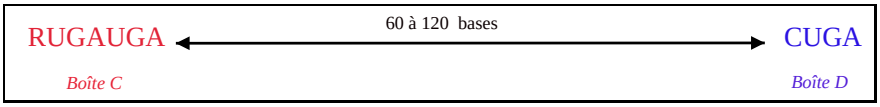


FIG. 3.7 – *Descripteur strict de PatScan pour rechercher les snoARN à boîtes C/D dans Saccharomyces cerevisiae.*

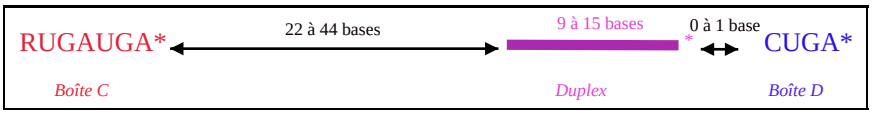


FIG. 3.8 – *Descripteur relâché de PatScan pour rechercher les snoARN à boîtes C/D dans Pyrococcus abyssi et Pyrobaculum aerophilum. Le symbole * signifie que l'on autorise une erreur.*

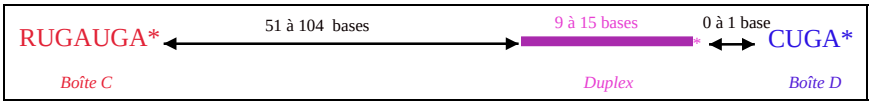


FIG. 3.9 – *Descripteur relâché de PatScan pour rechercher les snoARN à boîtes C/D dans Saccharomyces cerevisiae. Le symbole * signifie que l'on autorise une erreur dans le motif.*

`Param_1` et `Param_2` d'Erpin

Nous avons utilisé deux types de paramètres en prenant en compte les boîtes C, D, C' et D' ou seulement les boîtes C et D.

`Param_1` : Prise en compte pour la recherche des boîtes C, D, C' et D'. L'hélice terminale est seulement utilisée lorsque le génome entier est scruté.

`Param_2` : Prise en compte pour la recherche des boîtes C, D. L'hélice terminale est seulement utilisée lorsque le génome entier est scruté.

3.5.4 Paramétrage du logiciel spécifique

Le logiciel spécifique utilisé, Snoscan, prend en compte la recherche du duplex. Pour cela, il intègre en entrée un fichier de séquences correspondant aux cibles potentielles où l'interaction entre le candidat et la cible est recherchée.

Les différentes utilisations de Snoscan ont constitué à lui fournir différents types de données comme cibles potentielles : dans le cadre de la recherche de candidats dans les bases de données, nous avons utilisé parallèlement

- les ARNt + ARNr comme cibles potentielles pour les archae-bactéries et les ARNr pour la levure
- le génome entier avec les trois organismes.

La recherche de candidats sur le génome entier avec Snoscan a été effectuée avec les ARNr et/ou ARNt cibles seulement, la recherche de duplex sur le génome entier étant trop coûteuse en terme de temps.

3.5.5 Données de référence

Les jeux de données utilisés sont issus du génome d'une première archae-bactérie *Pyrococcus abyssi* [CBF+03]. La taille de ce génome archae-bactérien est de 1,76 Mbases et 59 gènes non codants sont annotés comme étant des snoARN [GCEB00, OLR+00].

Le second génome utilisé est celui de *Pyrobaculum aerophilum* avec une taille de 2,2 Mbases et 51 gènes non codants sont annotés comme étant des snoARN. Cette archae-bactérie a été séquencée en 2002 [FGLK+02].

Nous avons également utilisé le génome de *Sacharomyces cerevisiae* pour lequel 46 séquences de snoARN à boîtes C/D ont été identifiées [SF99]. Parmi ces 46 séquences, 41 sont données à la fois par Fournier et par Eddy et pour chacune d'entre elles, leur activité de méthylation a été mise en évidence ainsi que leur cible (l'ARNr `18S` ou l'ARNr `25S`). Les 5 autres ne sont décrites que par le site de Fournier [SF99], deux sont impliquées non pas dans un processus de méthylation mais dans un processus de clivage du précurseur de l'ARN ribosomique (`U3A` et`U3b`), 2 autres pour lesquelles l'activité et la cible n'ont pas été identifiées (`snR4` et `snR45`) et enfin une dernière où l'activité de méthylation n'est pas prouvée mais dont la cible serait identifiée.

3.5.5.1 Analyse des résultats obtenus avec Patscan et Erpin

Résultats pour les bases de données des snoARN des archae-bactéries : tables 3.14 et 3.15

Les 59 snoARN de *P. abyssi* sont trouvés dans la base de données avec Erpin en prenant en considération ou non les boîtes C' et D' (`Param_1` et `Param_2`). Dans le cas de PatScan, seul le descripteur strict trouve tous les vrais positifs. Le descripteur relâché (lorsque la présence des boîtes C' et D' n'est pas considérée, qu'une erreur est autorisée dans les boîtes C et D et que le duplex est recherché) ne trouve pas les 59 snoARN. Deux snoARN sont manqués car ils contiennent deux erreurs par rapport au consensus de la boîte C.

Les résultats avec *P. aerophilum* montre que seul le descripteur relâché autorise l'identification de tous les vrais positifs. Erpin, avec `Param_1` ainsi que PatScan avec le descripteur strict (dans les deux cas les boîte C' et D' sont considérées), présentent des solutions avec une très mauvaise sensibilité.

Résultats pour les génomes des archae-bactéries : tables 3.16 et 3.17

Les résultats obtenus en scrutant le génome complet des deux archae-bactéries montrent des résultats intéressants. Dans le cas du génome de *P. abyssi*, Erpin aussi bien que PatScan donnent des résultats avec une très bonne spécificité en tenant compte des boîtes C' et D' (respectivement `Param_1` et descripteur strict). Ces caractéristiques ne sont pas aussi spécifiques pour l'autre génome, *P. aerophilum* (16 et 7 vrais positifs respectivement). Une meilleur façon de trouver les snoARN pour *P. aerophilum* est de considérer seulement les boîtes C et D comme principales caractéristiques des snoARN. On obtient alors 51 vrais positifs avec le descripteur relâché de PatScan et 50 avec `Param_2` pour Erpin.

Résultats pour *S. cerevisiae* : tables 3.18 et 3.19

Remarquablement, les analyses des résultats de la base de données et de la séquence génomique montrent des résultats comparable avec ceux de *P. aerophilum*. En effet, lorsque seules les boîtes C et D sont décrites, les résultats obtenus avec `Param_2` pour Erpin et le descripteur

relâché pour PatScan, sont considérablement améliorés. Ce constat est en accord avec les données disponibles pour les snoARN de *S. cerevisiae* pour lesquels les boîtes C' et D' sont moins canoniques que celles trouvées chez *P. abyssi*.

La faible spécificité des résultats obtenus avec Erpin peut être expliquée par l'utilisation de l'alignement multiple des archae-bactéries (le seul disponible).

3.5.5.2 Analyse des résultats obtenus avec Snoscan

Les meilleurs résultats de Snoscan, lors d'une utilisation sur les bases de données, sont obtenus pour *S. cerevisiae*. Ceci est très certainement dû au fait que nous utilisons une version dédiée à la levure (les modèles d'apprentissage pour les archae-bactéries ne sont pas disponibles).

Les résultats obtenus avec Snoscan montrent un nombre important de faux positifs et quelques faux négatifs lors d'une recherche dans des banques de données de petits ARN de la levure. Parmi les 9 faux négatifs obtenus en utilisant comme jeux de données cibles les 3 ARNr de la levure, 4 séquences font partie des 5 séquences non référencées sur le site de Eddy[21] (activité ou cible non identifiée), 3 autres (`snR70, snR71` et `snR63`) sont mentionnées dans la documentation de Snoscan comme n'étant pas trouvées avec les paramètres par défaut et enfin deux snoARN appartenant au même groupe (*cluster*) (`snR57` et `snR55`). L'ajout des ARN de transfert aux groupes de séquences cible permet de trouver de nouveaux candidats, mais sans augmenter le nombre de vrais positifs. Parmi les 41 snoARN bien identifiés, aucun n'est cité comme interagissant avec un ARN de transfert.

De façon surprenante, lorsque les positions méthylées sur les ARNr cibles (fournies par les auteurs) sont fournies à Snoscan, on observe un effet important sur la fonction de score, diminuant le nombre de faux négatifs et augmentant celui des vrais positifs. Les 3 nouveaux snoARN identifiés sont `snR55`, `snR57` et `snR71`. Le nombre de faux positifs augmente de façon importante. Cet accroissement est directement lié à la pondération positive que le programme doit faire lorsque le site de méthylation est donné et cette amélioration du score intermédiaire permet dans certains cas au programme de trouver des candidats avec une boîte C plus dégénérée.

L'utilisation du génome entier comme cible potentielle permet à Snoscan de trouver des cibles pour l'ensemble des snoARN de la base de données à l'exception de `U3a`. Ces résultats sont intéressants dans la mesure où ils proposent de nouvelles cibles pour les snoARN orphelins (snoARN dont l'existence a été prouvée expérimentalement mais pour qui on ignore la cible). Cependant, le nombre de solutions obtenue rend délicate l'analyse des résultats.

21. http://rna.wustl.edu/snoRNAdb/

	Solutions		**Vrai**	**Faux**	**Faux**
	unique	*double*	**positifs**	**négatifs**	**positifs**
PatScan					
	66				
Descripteur strict	*61*	*5*	59	0	5
	79				
Descripteur relâché	*79*	*0*	57	2	22
Erpin					
	59				
`Param_1`	*59*	*0*	59	0	0
	59				
`Param_2`	*59*	*0*	59	0	0
Snoscan					
	31				
cible : ARNr + ARNt	*20*	*11*	20	39	11
	1016				
cible : génome complet	*46*	*970*	46	13	970

TAB. 3.14 – *Résultats pour les 59 snoARN de Pyrococcus abyssi.*

	Solutions		**Vrai**	**Faux**	**Faux**
	unique	*double*	**positifs**	**négatifs**	**positifs**
PatScan					
	8				
Descripteur strict	*7*	*0*	7	44	1
	78				
Descripteur relâché	*78*	*0*	51	0	26
Erpin					
	38				
`Param_1`	*38*	*0*	38	13	0
	50				
`Param_2`	*50*	*0*	50	1	0
Snoscan					
	19				
cible : ARNr + ARNt	*19*	*0*	10	41	9
	806				
cible : génome complet	*45*	*771*	45	6	771

TAB. 3.15 – *Résultats pour les 51 snoARN de Pyrobaculum aerophilum.*

	Solutions		**Vrai positifs**	**Faux négatifs**	**Faux positifs**
	unique	*double*			
PatScan					
Descripteur strict	267 *253*	*14*	59	0	208
Descripteur relâché	12647 *12647*	*0*	57	2	12590
Erpin					
`Param_1`	64 *64*	*0*	59	0	5
`Param_2`	10359 *10359*	*0*	59	0	10300
Snoscan					
cible : ARNr + ARNt	4275 *1611*	*2664*	27	32	4248

TAB. 3.16 – *Résultats obtenus avec le génome de Pyrococcus abyssi lors de la recherche de snoARN.*

	Solutions		**Vrai positifs**	**Faux négatifs**	**Faux positifs**
	unique	*double*			
PatScan					
Descripteur strict	63 *63*	*0*	7	44	46
Descripteur relâché	8787 *8787*	*0*	51	0	8736
Erpin					
`Param_1`	16 *16*	*0*	16	35	0
`Param_2`	277 *277*	*0*	50	1	227
Snoscan					
cible : ARNr + ARNt	2654 *136*	*2518*	10	41	2644

TAB. 3.17 – *Résultats obtenus avec le génome de Pyrobaculum aerophilum lors de la recherche de snoARN.*

	Solutions		**Vrai positifs**	**Faux négatifs**	**Faux positifs**
	unique	*double*			
PatScan					
	114				
Descripteur strict	*114*	*0*	46	0	68
	112				
Descripteur relâché	*112*	*0*	46	0	6
Erpin					
	19				
`Param_1`	*19*	*0*	19	27	0
	47				
`Param_2`	*47*	*0*	46	0	1
Snoscan					
	200				
cible : ARNr	*37*	*163*	37	9	163
cible :ARNr	271				
positions des sites methylation	*40*	*231*	40	6	231
	153119				
cible : génome	*45*	*153044*	45	1	153044

TAB. 3.18 – *Résultats pour les 46 snoARN de Saccharomyces cerevisiae.*

	Solutions		**Vrai positifs**	**Faux négatifs**	**Faux positifs**
	unique	*double*			
PatScan					
	289468				
Descripteur strict	*289468*	*0*	46	0	289422
	274579				
Descripteur relâché	*274579*	*0*	46	0	274533
Erpin					
	127				
`Param_1`	*127*	*0*	0	46	127
	63547				
`Param_2`	*63547*	*0*	31	15	63516
Snoscan					
	1687				
cible : ARNr	*551*	*1136*	36	10	1651
cible : ARNr	5002				
positions des sites methylation	*2701*	*2301*	36	10	4966

TAB. 3.19 – *Résultats pour le génome de Saccharomyces cerevisiae.*

3.6 Analyse approfondie des résultats de la recherche de snoARN dans la séquence de *P. aerophilum*

3.6.1 Objectif

Les résultats obtenus dans l'étude présentée ci-dessus, autour de la recherche de snoARN chez les archae-bactéries, nous ont interrogé quant à l'existence potentielle de nouveaux snoARN dans le génome de *P. aerophilum*, ceux-ci n'ayant pas fait l'objet d'une publication autre que l'annotation disponible au NCBI pour ce génome et la revue sur les archae-bactéries présentée dans [FGLK+02]. Nous avons donc plus particulièrement regardé ces candidats dans les régions inter-géniques de ce génome. Pour effectuer nos recherches, nous avons utilisé PatScan avec un descripteur relâché (double utilisation de PatScan pour rechercher des interactions inter-moléculaires entre les candidats et les ARNr ou ARNt de *P. aerophilum*).

3.6.2 Extraction des régions inter-géniques

Nous avons extrait du génome les régions inter-géniques en tenant compte de l'annotation disponible pour ce génome sur le site du NCBI. A chaque région intergénique, nous avons ajouté de part et d'autre, 20 nucléotides, afin de gérer les chevauchements potentiels aux bords. Pour chaque région intergénique ainsi construite, nous avons ensuite cherché, à l'aide du descripteur relâché, les candidats potentiels.

3.6.3 Sélection des candidats

Pour chaque candidat retenu, nous avons recherché une cible potentielle (parmi les ARNr 16S, ARNr 23S, ARNr 5S, et 48 ARNt) localisée en amont de la boîte D. Pour cela, nous avons cherché à former avec une cible potentielle, à partir d'un nucléotide en amont de la boîte D, un duplex de 9 à 15 paires de bases de type GC, CG, AU, UA, contenant une erreur au plus.

3.6.4 Résultats

Le nombre de candidats trouvés dans les régions intergéniques est 1191. Parmi ces candidats, deux snoARN sont manquants car ils recouvrent largement des régions annotées comme étant codantes. Parmi ces 1191 candidats, 303 contiennent une région pouvant former un duplex avec au moins une cible. Pour certains plusieurs cibles sont possibles (1710 résultats pour 303 candidats).

Nous avons examiné ces 303 candidats afin de retenir ceux qui nous paraissaient les plus pertinents. Les candidats retenus sont présentés dans la figure 3.10.

Cette sélection a été réalisée en tenant compte de :

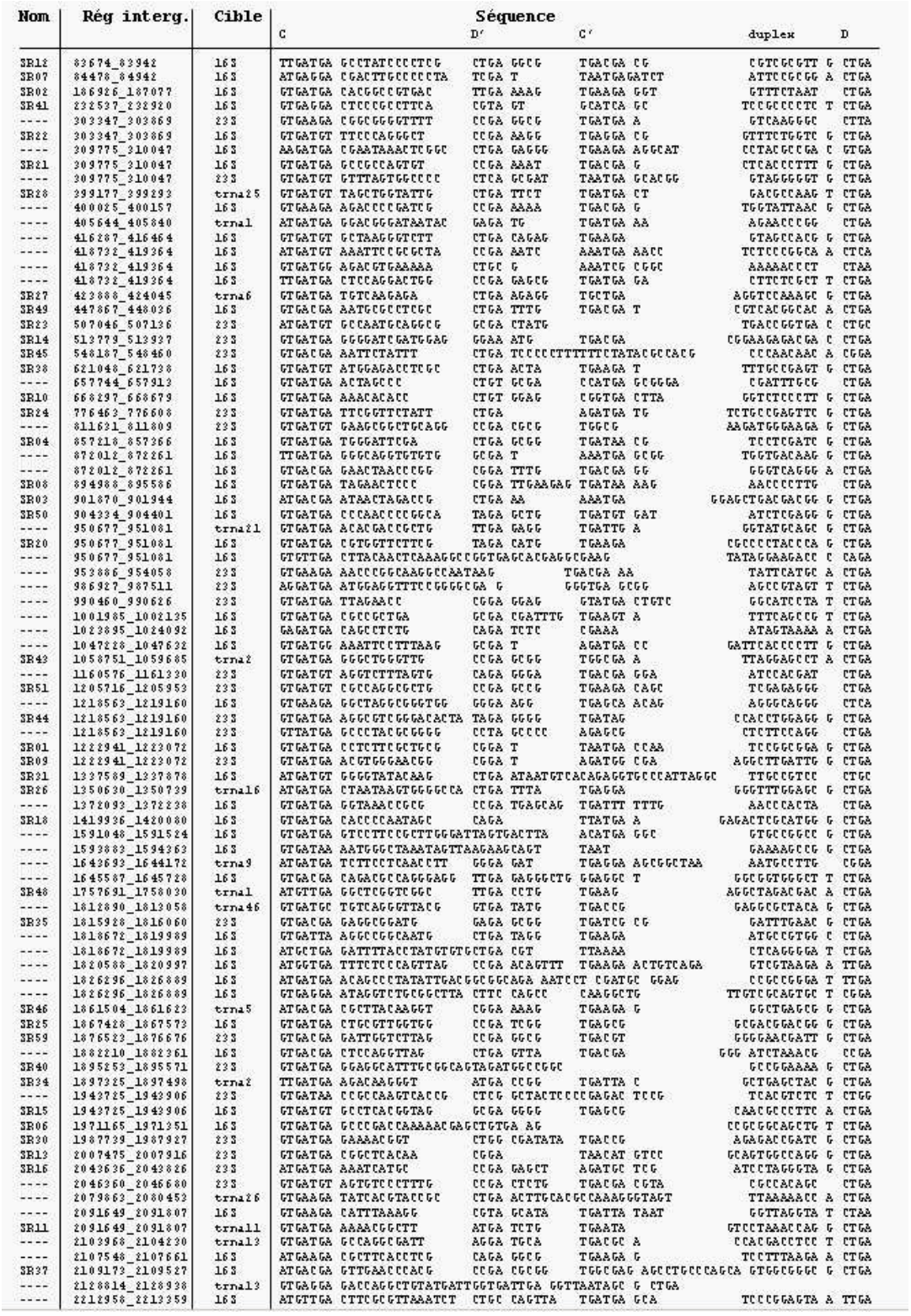
– La canonicité des boîtes C et D. Lorsque ces deux boîtes coexistent, et qu'une région cible est localisée, le candidat est retenu. Nous avons retenu ce critère car il semble caractériser une majorité de snoARN chez les archae-bactéries. Quelques candidats présentent ce critère. Par contre, bon nombre de candidats contiennent les 4 boîtes dont au moins une est dégénérée.

- La présence des quatre boîtes C, D, C' et D', même dégénérées. La majorité des candidats peuvent se classer dans cette catégorie. Pour certains candidats, cependant, nous n'avons pas pu identifier clairement (ou pas du tout) soit la boîte C', soit la boîte D'.
- La taille du duplex formé. Nous avons retenu certains candidats contenant des boîtes dégénérées mais dont la région pouvant s'apparier à la cible comportait plus de 12 paires de bases (10 en moyenne chez les archae-bactéries).
- Une possible organisation en groupe (*cluster*). L'organisation en groupe n'est pas une caractéristique mise en évidence chez les archae-bactéries [GCEB00, OLR$^+$00] alors que bon nombre de snoARN sont organisés en groupe chez les plantes [BCL$^+$01, QMZ$^+$01]. Les résultats obtenus dans le cadre de cette analyse montre que ce critère peut être retenu dans le cadre du génome de *P. aerophilum* (11 groupes potentiels).

Nous avons ainsi pu mettre en évidence avec Patscan:

- 42 snoARN parmi les 51 snoARN annotés sur ce génome. 11 snoARN sont manquant pour les raisons suivantes : soit ils chevauchent des régions annotées comme codantes (2 candidats), soit ils ne présentent pas en amont de la boîte D une région pouvant s'apparier sous la forme ci-dessus avec une des cibles utilisées. Ces candidats sont présentés figure 3.10 et portent le nom qui leur est affecté sur le site du NCBI. Sur chaque ligne sont présentés : le nom du candidat lorsqu'il est connu, les positions de la région intergénique le contenant, un des ARN cible, la séquence décomposée pour mettre en évidence les boîtes C, D', C', D' ainsi que le duplex pouvant être formé avec l'ARN cible.
- Environ 40 snoARN nouveaux potentiels. Ces candidats sont présentés dans la figure 3.10.

Il ne nous est pas possible à l'heure actuelle, de confirmer les candidats proposés. Une validation expérimentale est en cours (A. Mougin, IBCG/CNRS, Toulouse).



Nom	Rég interg.	Cible	Séquence C	D′	C′	duplex	D
SR12	83674_83942	16S	TTGATGA GCCTATCCCCTCG	CTGA GGCG	TGACGA CG	CGTCGCGTT G	CTGA
SR07	84478_84942	16S	ATGAGGA CGACTTGCCCCCTA	TCGA T	TAATGAGATCT	ATTCCGCGG A	CTGA
SR02	186926_187077	16S	GTGATGA CACGGCCGTGAC	TTGA AAAG	TGAAGA GGT	GTTTCTAAT	CTGA
SR41	232527_232920	16S	GTGAGGA CTCCCGCCTTCA	CGTA GT	GCATCA GC	TCCGCCCCTC T	CTGA
----	303347_303869	23S	GTGAAGA CGGCGGGGTTTT	CCGA GGCG	TGATGA A	GTCAAGGGC	CTTA
SR22	303347_303869	16S	GTGATGT TTCCCAGGGCT	CCGA AAGG	TGAGGA CG	GTTTCTGGTC G	CTGA
----	309775_310047	16S	AAGATGA CGAATAAACTCGGC	CTGA GAGGG	TGAAGA AGGCAT	CCTACGCCGA C	GTGA
SR21	309775_310047	16S	GTGATGA GCCGCCAGTGT	CCGA AAAT	TGACGA G	CTCACCCTTT G	CTGA
----	309775_310047	23S	GTGATGT GTTTAGTGGCCCC	CTCA GCGAT	TAATGA GCACGG	GTAGGGGGT G	CTGA
SR28	399177_399293	trna25	GTGATGT TAGCTGGTATTG	CTGA TTCT	TGATGA CT	GACGCCAAG T	CTGA
----	400025_400157	16S	GTGAAGA AGACCCCGATCG	CCGA AAAA	TGACGA G	TGGTATTAAC G	CTGA
----	405644_405840	trnal	ATGATGA GGACGGGATAATAC	GAGA TG	TGATGA AA	AGAACCCGG	CTGA
----	416287_416464	16S	GTGATGT GCTAAGGGTCTT	CTGA CAGAG	TGAAGA	GTAGCCACG G	CTGA
----	418732_419364	16S	ATGATGT AAATTCCGCGCTA	CCGA AATC	AAATGA AACC	TCTCCCGGCA A	CTCA
----	418732_419364	16S	GTGATGG AGACGTGAAAAA	CTGC G	AAATCG CGGC	AAAAACCCT	CTAA
----	418732_419364	16S	TTGATGA CTCCAGGACTGG	CCGA GAGCG	TGATGA GA	CTTCTCGCT T	CTGA
SR27	423888_424045	trna6	GTGATGA TGTCAAGAGA	CTGA AGAGG	TGCTGA	AGGTCCAAAGC G	CTGA
SR49	447867_448036	16S	GTGACGA AATGCGCCTCGC	CTGA TTTG	TGACGA T	CGTCACGGCAC A	CTGA
SR23	507046_507136	23S	ATGATGT GCCAATGCAGGCG	GCGA CTATG		TGACCGGTGA C	CTGC
SR14	513779_513937	23S	GTGATGA GGGGATCGATGGAG	GGAA ATG	TGACGA	CGGAAGAGACGA C	CTGA
SR45	548187_548460	23S	GTGACGA AATTCTATTT	CTGA TCCCCCTTTTTTCTATACGCCACG		CCCAACAAC A	CGGA
SR38	621048_621738	16S	GTGATGT ATGGAGACCTCGC	CTGA ACTA	TGAAGA T	TTTGCCGAGT G	CTGA
----	657744_657913	16S	GTGATGA ACTAGCCC	CTGT GCGA	CCATGA GCGGGA	CGATTTGCG	CTGA
SR10	668297_668679	16S	GTGATGA AAACACACC	CTGT GGAG	CGGTGA CTTA	GGTCTCCCTT G	CTGA
SR24	776463_776608	23S	GTGATGA TTCGGTTCTATT	CTGA	AGATGA TG	TCTGCCGAGTTC G	CTGA
----	811631_811809	23S	GTGATGT GAAGCGGGTGCAGG	CCGA CGCG	TGGCG	AAGATGGGAAGA G	CTGA
SR04	857218_857366	16S	GTGATGA TGGGATTCGA	CTGA GCGG	TGATAA CG	TCCTCGATC G	CTGA
----	872012_872261	16S	TTGATGA GGGCAGGTGTGTG	GCGA T	AAATGA GCGG	TGGTGACAAG G	CTGA
----	872012_872261	16S	GTGACGA GAACTAACCCGG	CGGA TTTG	TGACGA GG	GGGTCAGGG A	CTGA
SR08	894988_895586	16S	GTGATGA TAGAACTCCC	CGGA TTGAAGAG	TGATAA AAG	AACCCCTTG	CTGA
SR03	901870_901944	16S	ATGACGA ATAACTAGACCG	CTGA AA	AAATGA	GGAGCTGACGACGG G	CTGA
SR50	904324_904401	16S	GTGATGA CCCAACCCCGGCA	TAGA GCTG	TGATGT GAT	ATCTCGAGG G	CTGA
----	950677_951081	trna21	GTGATGA ACACGACCGCTG	TTGA GAGG	TGATTG A	GGTATGCAGC G	CTGA
SR20	950677_951081	16S	GTGATGA CGTGGTTCTTCG	TAGA CATG	TGAAGA	CGCCCCTACCCA G	CTGA
----	950677_951081	16S	GTGTTGA CTTACAACTCAAAGGCCGGTGAGCACGAGGCGAAG			TATAGGAAGACC C	CAGA
----	953886_954058	23S	GTGAAGA AACCCGGCAAGGCCAATAAG	TGACGA AA		TATTCATGC A	CTGA
----	986927_987511	23S	AGGATGA ATGGAGGTTTCCGGGGCGA G	GGGTGA GCGG		AGCCGTAGT T	CTGA
----	990460_990626	23S	GTGATGA TTAGAACC	CGGA GGAG	GTATGA CTGTC	GGCATCCTA T	CTGA
----	1001985_1002135	16S	GTGATGA CGCCGCTGA	GCGA CGATTTG	TGAAGT A	TTTCAGCCG T	CTGA
----	1023895_1024092	16S	GAGATGA CAGCCTCTG	CAGA TCTC	CGAAA	ATAGTAAAA A	CTGA
----	1047228_1047632	16S	GTGATGG AAATTCCTTTAAG	GCGA T	AGATGA CC	GATTCACCCCTT G	CTGA
SR42	1058751_1059685	trna2	GTGATGA GGGCTGGGTTG	CCGA GCGG	TGGCGA A	TTAGGAGCCT A	CTGA
----	1160576_1161230	23S	GTGATGT AGGTCTTTAGTG	CAGA GGGA	TGACGA GGA	ATCCACGAT	CTGA
SR51	1205716_1205953	23S	GTGATGT CGCCAGGCGCTG	CCGA GCCG	TGAAGA CAGC	TCGAGAGGG	CTGA
----	1218563_1219160	16S	GTGAAGA GGCTAGGCGGGTGG	GGGA AGG	TGAGCA ACAG	AGGGCAGGG	CTCA
SR44	1218563_1219160	23S	GTGATGA AGGCGTCGGGACACTA TAGA GGGG		TGATAG	CCACCTGGAGG G	CTGA
----	1218563_1219160	23S	GTTATGA GCCCTACGCGGGG	CCTA GCCCC	AGAGCG	CTCTTCCAGG	CTGA
SR01	1222941_1223072	16S	GTGATGA CCTCTTCGCTGCG	CGGA T	TAATGA CCAA	TCCGGCGGA G	CTGA
SR09	1222941_1223072	23S	GTGATGA ACGTGGGAACGG	CGGA T	AGATGG CGA	AGGCTTGATTG G	CTGA
SR31	1337589_1337878	16S	ATGATGT GGGGTATACAAG	CTGA ATAATGTCACAGAGGTGCCCATTAGGC		TTGCCGTCC	CTGC
SR26	1350630_1350729	trna16	ATGATGA CTAATAAGTGGGGCCA CTGA TTTA	TGAGGA		GGGTTTGGAGC G	CTGA
----	1372093_1372238	16S	GTGATGA GGTAAACCGCG	CCGA TGAGCAG	TGATTT TTTG	AACCCACTA	CTGA
SR18	1419936_1420080	16S	GTGATGA CACCCCAATAGC	CAGA	TTATGA A	GAGACTCGCATGG G	CTGA
----	1591048_1591524	16S	GTGATGA GTCCTTCCGCTTGGGATTAGTGACTTA		ACATGA GGC	GTGCCGGCC G	CTGA
----	1593882_1594362	16S	GTGATAA AATGGGCTAAATAGTTAAGAAGCAGT		TAAT	GAAAAGCCG G	CTGA
----	1643693_1644172	trna9	ATGATGA TCTTCCTCAACCTT GGGA GAT		TGAGGA AGCGGCTAA	AATGCCTTG	CGGA
----	1645587_1645728	16S	GTGACGA CAGACGCCAGGGAGG TTGA GAGGGCTG		GGAGGC T	GGCGGTGGGCT T	CTGA
SR48	1757691_1758030	trnal	ATGTTGA GGCTCGGTCGGC TTGA CCTG		TGAAG	AGGCTAGACGAC A	CTGA
----	1812890_1813058	trna46	GTGATGC TGTCAGGGTTACG GTGA TATG		TGACCG	GAGGCGCTACA G	CTGA
SR35	1815928_1816060	23S	GTGACGA GAGGCGGATG	GAGA GCGG	TGATCG CG	GATTTGAAC G	CTGA
----	1818672_1819989	16S	GTGATTA AGGCCGGCAATG	CTGA TAGG	TGAAGA	ATGCCGTGG C	CTGA
----	1818672_1819989	16S	ATGCTGA GATTTTACCTATGTGTGCTGA CGT		TTAAAA	CTCAGGGGA T	CTGA
----	1820588_1820997	16S	ATGGTGA TTTCTCCCAGTTAG CCGA ACAGTTT		TGAAGA ACTGTCAGA	GTCGTAAGA A	TTGA
----	1826296_1826889	16S	ATGATGA ACAGCCCTATATTGACGGCGGCAGA AATCCT CGATGC GGAG			CCGCCCGGA T	TTGA
----	1826296_1826889	16S	GTGAGGA ATAGGTCTGCGGCTTA CTTC CAGCC		CAAGGCTG	TTGTCGCAGTGC T	CGGA
SR46	1861504_1861623	trna5	ATGACGA CGCTTACAAGGT CGGA AAAG		TGAAGA G	GGCTGAGCG G	CTGA
SR25	1867428_1867573	16S	GTGATGA CTGCGTTGGTGG CCGA TCGG		TGAGCG	GCGACGGACGG G	CTGA
SR59	1876523_1876676	23S	GTGACGA GATTGGTCTTAG CCGA GGCG		TGACGT	GGGGAACGATT G	CTGA
----	1882210_1882361	16S	GTGACGA CTCCAGGTTAG CTGA GTTA		TGACGA	GGG ATCTAAACG	CCGA
SR40	1895253_1895571	23S	GTGATGA GGAGGCATTTGCGGCAGTAGATGGCCGGC			GCCGGAAAA G	CTGA
SR34	1897225_1897498	trna2	TTGATGA AGACAAGGGT ATGA CCGG		TGATTA C	GCTGAGCTAC G	CTGA
----	1943725_1943906	23S	GTGATAA CCGCCAAGTCACCG CTCG GCTACTCCCCGAGAC TCCG			TCACGTCTC T	CTGG
SR15	1943725_1943906	16S	GTGATGT GCCTCACGGTAG GCGA GGGG		TGAGCG	CAACGCCCTTC A	CTGA
SR06	1971165_1971351	16S	GTGATGA GCCCGACCAAAAACGAGCTGTGA AG			CCGCGGCAGCTG T	CTGA
SR30	1987739_1987927	23S	GTGATGA GAAAACGGT CTGG CGATATA		TGACCG	AGAGACCGATC G	CTGA
SR13	2007475_2007916	23S	GTGATGA CGGCTCACAA CGGA		TAACAT GTCC	GCAGTGGCCAGG G	CTGA
SR16	2043636_2043826	23S	ATGATGA AAATCATGC CCGA GAGCT		AGATGC TCG	ATCCTAGGGTA G	CTGA
----	2046360_2046680	23S	GTGATGT AGTGTCCCTTTG CCGA CTCTG		TGACGA CGTA	CGCCACAGC	CTGA
----	2079663_2080453	trna26	GTGAAGA TATCACGTACCGC CTGA ACTTGCACGCCAAAGGGTAGT			TTAAAAACC A	CTGA
----	2091649_2091807	16S	GTGAAGA CATTTAAAGG CGTA GCATA		TGATTA TAAT	GGTTAGGTA T	CTAA
SR11	2091649_2091807	trnall	GTGATGA AAAACGGCTT ATGA TCTG		TGAATA	GTCCTAAACCAG G	CTGA
----	2103968_2104230	trnal3	GTGATGA GCCAGGCGATT AGGA TGCA		TGACGC A	CCACGACCTCC T	CTGA
----	2107548_2107661	16S	ATGAAGA CGCTTCACCTG CAGA GGCG		TGAAGA G	TCCTTTAAGA A	CTGA
SR37	2109173_2109527	16S	ATGACGA GTTGAACCCACG CCGA CGCGG		TGGCGAG AGCCTGCCCAGCA GTGGCGGGC G		CTGA
----	2128814_2128938	trnal3	GTGAGGA GACCAGGCTGTATGATTGGTGATTGA GGTTAATAGC G CTGA				
----	2212958_2213359	16S	ATGTTGA CTTCGCGTTAAATCT CTGC CAGTTA		TGATGA GCA	TCCCGGAGTA A	TTGA

FIG. 3.10 – *Une quarantaine de nouveaux snoARN potentiels à boîtes C/D pour P. aerophilum. Les noms donnés en pointillés désignent les nouveaux candidats potentiels.*

3.7 Discussion

Avantages de chaque famille de logiciels

Les logiciels spécifiques semblent être de meilleurs outils pour rechercher les ARN dont la signature est réellement caractéristique de la famille considérée. Par exemple, dans le cas de la recherche des ARN de transfert dans le génome de *E. coli*, quasiment tous les logiciels spécifiques trouvent l'ensemble des vrais positifs (à l'exception de l'ARNt de la sélénocystéine qui ne possède par une signature canonique et pour lequel un protocole spécifique de recherche a été écrit). Parmi ces logiciels spécifiques, les meilleurs résultats sont montrés par tRNAscan-SE et Aragorn.

Cependant, les logiciels généralistes présentent d'intéressants avantages dans les cas où la structure du motif n'a pas été finement décrite. En effet, ces logiciels différencient clairement la description du motif et les algorithmes de recherche. Ils proposent un langage pour décrire les ARN à rechercher. Ainsi, l'utilisateur dispose d'une interface pour modéliser l'ARN auquel il s'intéresse sans avoir besoin pour cela de programmer. Ces langages sont généralement assez simples, comme dans le cas de RnaMot.

Le langage de PatScan autorise la description de différents types d'erreurs (insertion, deletion ou mésappariement) pour la description de mots ou d'hélices, rendant ainsi possible la description d'hélices contenant des renflements ou des boucles internes. Lors de notre analyse comparative, il a été possible en utilisant PatScan de spécifier un descripteur relâché pour les snoARN avec la signature des séquence consensus correspondant aux deux boîtes C et D, en autorisant une erreur sur la description de chacune, alors qu'avec RnaMot, ceci est impossible (autoriser des erreurs pour la recherche de mots). Ainsi nous avons trouvé que PatScan peut offrir un très bon compromis entre simplicité et efficacité.

Un autre avantage des outils généralistes est que l'on peut facilement modifier une description pour effectuer une nouvelle recherche. Dans le cas des logiciels spécifiques, il est impossible de modifier aussi aisément la description d'un motif et de l'algorithme de recherche, les deux étant étroitement liés.

Les logiciels généralistes sont souvent plus efficaces et adaptés pour rechercher un motif peu défini dans le but d'examiner ensuite plus en détails les résultats obtenus. Ainsi, ils peuvent être utilisés pour une pré-recherche et fournir dans un second temps un ensemble de candidats à évaluer à des logiciel plus dédiés à un type de molécule. Cette idée a été mise en pratique dans tRNAscan-SE en combinant l'utilisation de Cove avec les versions relâchées (donc plus sensibles) de tRNAscan et Pol3Scan.

Exhaustivité des logiciels généralistes

Les résultats obtenus à partir des logiciels généralistes montrent l'importance de considérer les régions chevauchantes en trouvant les vrais positifs même si le nombre de solutions augmente considérablement lorsque l'on vise l'exhaustivité pure. En effet, RnaBob manque 17 ARN de transfert candidats sur le génome de *S. cerevisiae* lorsque les solutions chevauchantes ne sont pas considérées. Selon le logiciel généraliste, les régions chevauchantes ne sont pas considérées de la même façon pour l'exploitation des résultats:

RnaMot présente les résultats chevauchants dans un fichier séparé.

RnaMotif et Palingol proposent en annexe des fonctions pour extraire les résultats se chevauchant.

RnaBob ne donne pas les solutions qu'il considère comme des alternatives.

Une autre façon d'analyser un grand nombre de solutions pourrait être de les regrouper selon différents critères (i) positions initiale et finale identiques, ou (ii) position initiale identique et candidat contenu dans un autre.

Une autre façon de représenter un grand nombre de solutions pourrait consister à regrouper les candidats se chevauchant sur toute une région de la séquence génomique en conservant seulement les extrémités correspondants à la plus petite valeur pour la position initiale et à la plus grande valeur pour la position finale de la région couverte. Dans le cas de la recherche des ARNt sur le génome de *S.cerevisiae* (804044 candidats), 64990 groupes ont ainsi été obtenus, diminuant le nombre de solutions d'un facteur multiplicatif de 12. Pour *P. abyssi*, les 267 candidats snoARN trouvés avec PatScan appartiennent à 191 groupes.

Compromis entre les deux familles de logiciels?

Les logiciels généralistes travaillant à partir d'alignements (Cove et Erpin) apparaissent comme le meilleur compromis entre les outils généralistes et les logiciels spécifiques. Les problèmes inhérents à l'utilisation de ces programmes concernent le nombre de séquences nécessaires dans l'alignement pour la phase d'apprentissage (le nombre doit être suffisant) et la qualité de l'alignement multiple. Dans le cas des ARNt, pour lesquels les logiciels spécifiques tels que tRNAscan-SE ou Aragorn, l'utilisation de logiciels spécifiques est le meilleur choix à faire pour un biologiste. Par contre, le cas des snoARN représente un exemple de famille de motifs pour laquelle l'utilisateur peut rapidement construire une signature plus spécifique avec les outils généralistes.

Où effectuer la recherche?

Dans cette étude les génomes complets sont scrutés. En effet, effectuer une recherche dans les régions inter-géniques est dépendant de la qualité de l'annotation et certains candidats pourraient être manqués si une région n'est pas bien annotée. Par exemple, la recherche d'ARN de transfert à partir des régions inter-géniques de *E. coli* a donné seulement 78 vrais positifs avec tRNAscan-SE au lieu de 88. 10 vrais positifs sont ainsi manqués car ils chevauchent des régions annotées comme gènes codants pour des protéines. Dans le cas de l'analyse fouillée des résultats de *P. aerophilum* deux candidats ont été ratés du fait de leur présence dans des régions annotées codantes.

Les faux positifs?

Un dernier point à discuter concerne les candidats considérés comme faux positifs.

- Sont-ils des pseudo-gènes?
- Sont-ils fonctionnels?

Il est impossible de répondre à cette question sans utiliser par exemple des expériences de *microarrays* où des milliers de gènes peuvent être simultanément testés. Dans *E. coli*, 545 candidats ont été trouvés avec RnaMot sur le génome complet. 241 candidats, représentant moins de la

moitié des solutions, ont été trouvés dans les régions inter-géniques (12% du génome complet), 166 candidats ont été trouvés dans les régions codantes (88 % du génome) et 138 sont chevauchant avec ces deux types de régions. Dans le génome de *P. aerophilum* nous avons pu proposer environ 40 nouveaux snoARN dont quelques uns nous paraissent certains.

3.8 Conclusion

Les conclusions à extraire de cette analyse comparative prennent différentes directions.

- La plupart des logiciels généralistes offrent une façon rapide de décrire un motif ARN donné, ils permettent à l'utilisateur de rapidement effectuer des recherches sur des séquences génomiques. De plus, le motif peut rapidement être modifié pour le rendre plus ou moins sensible selon le niveau de connaissances dont dispose l'utilisateur. De telles modifications ne sont pas facilement envisageable avec un logiciel spécifique. Cependant, les profils et les représentations probabilistes des motifs ont montré leur utilité pour obtenir une meilleur spécificité. De plus, ils offrent une façon d'évaluer et de grouper les candidats mais ils sont peu efficaces en terme de temps d'exécution. Une combinaison des deux approches a clairement montré son intérêt avec l'exemple de tRNAscan-SE où les candidats obtenus à partir des versions relâchées de deux logiciels spécifiques sont ensuite évalués par le modèle probabiliste offert par Cove.
- Les outils généralistes diffèrent aussi entre eux selon leur langage et la façon de présenter leurs résultats. Les langages de ces logiciels sont en général flexibles et d'utilisation simple.
- Un challenge futur pour les logiciels généralistes concerne la façon de décrire les nouvelles générations d'ARN:
  - permettant les signatures probabilistes,
  - la spécification de complexes pouvant être formés par les interactions antisens entre différentes régions d'un génome,
  - des fonctions de score plus sophistiquées et diversifiées.
- Le grand nombre de solutions obtenues par les logiciels généralistes peut aussi donner un axe de recherche. Ces prédictions sont-elles fonctionnelles? Une façon de répondre à cette question serait de développer des programmes informatiques capables de grouper ensemble les solutions. Une fonction regroupant les solutions satisfaisant l'inclusion ou les chevauchements diminuent le nombre de solutions d'un facteur 12 pour *S. cerevisiae*. Une autre façon pour aider l'analyse d'un grand nombre de solutions est de les représenter sur le génome annoté en regroupant les régions candidates selon leur localisation structurale génomique. Finalement, l'accent devra être mis sur les grands ensembles de faux positifs par des expériences à grande échelle.

3.9 Bilan et objectifs

Cette analyse de logiciels nous a permis de mettre en avant un ensemble de points qui nous paraissent intéressants ou essentiels pour les logiciels généralistes de recherche de motifs. Ces points portent sur la possibilité de spécifier notamment :

- des éléments de séquence (mots) et de structure secondaire (hélices) de base non exacts;
- des éléments structurés tels que triple hélices, quadruple hélices ou pseudo-noeud (la plupart des logiciels généralistes) pour les ARN;
- des contraintes sur le contenu d'un élément (mot ou hélice);
- la pondération d'un élément ou d'un ensemble d'éléments de motif dans le cadre d'une fonction de score globale;
- des contraintes de distance;
- des règles d'appariement et le contrôle du nombre d'appariements d'un type donné.

Suite à cette analyse, les aspects que nous souhaiterions pouvoir intégrer concernent les points suivants :

- la spécification des interactions entre molécules. Les snoARN sont un exemple du type d'interaction entre deux molécules que nous souhaitons pouvoir modéliser avec notre formalisme. La recherche de ces petits ARN sur la base de leur seule structure individuelle génère, quelque soit le logiciel utilisé, un nombre important de faux positifs. Il est donc impératif dans une deuxième étape de la recherche de candidats d'éliminer un certain nombre d'entre eux sur le critère de la présence du duplex pouvant se former avec une région d'un ARN connu comme étant une cible potentielle. Actuellement, aucun outil généraliste ne permet de définir cette relation intermoléculaire entre le petit ARN nucléolaires et une cible potentielle. De plus, un nombre croissant de candidats orphelins (sans cible) apparaît. Il semble donc nécessaire d'incorporer dans la recherche de motifs un moyen générique de spécifier des interactions possibles avec d'autres molécules.
- la possibilité de spécifier la présence "optionnelle" d'un ou plusieurs éléments de motif. Les petits ARN antisens à boîtes C/D représentent aussi un bon exemple de ce type de motif ou la présence d'une hélice terminale conforte l'identification du candidat comme étant un snoARN à boîtes C/D.
- la possibilité d'adjoindre des notions d'approximation aux recherches à effectuer : (ex: rechercher le plus long suffixe ou préfixe d'un mot), mais aussi en terme d'expression en offrant la possibilité d'utiliser des clauses du type : riche en GC..., ne contient pas tel motif structuré (opérations de négation)... Des logiciels tels que Palingol et RnaMotif permettent ce genre de contrôle dans une fonction de score.

3.9.1 Objectifs dans le cadre de la thèse

Un des objectifs de la thèse porte sur la proposition d'un modèle et les développements informatiques d'un outil généraliste permettant de rechercher dans les séquences génomiques des régions non codantes fonctionnelles pouvant mettre en jeu des interactions intra et/ou intermoléculaires. L'étude des logiciels généralistes présentées précédemment nous a permis de mettre en avant un certain nombre de points pour nos développements.

3.9.1.1 Les variables

Les variables représentent les unités élémentaires du motif. Elles peuvent être de deux types selon que les éléments de structure sont considérés de manière spécifique ou non. Le domaine d'une variable est donné par la liste des éléments de séquence ou de structure qu'il est possible de trouver dans une séquence. Dans la formalisation proposée par Eidhammer *al* [EGJR01], les variables sont associées aux éléments de séquence. Après un filtrage local sur les éléments de structure, une reformulation du problème relie ensuite, au moyen de variables qui représentent les contraintes d'appariement, deux variables qui constituent un élément de structure. Dans la plupart des outils existants (Palingol excepté), on peut relier aisément chaque élément du motif à la formulation proposée par Eidhammer. En effet, des outils comme RnaBob, RnaMot, PatScan, PatSearch exprime de manière linéaire un motif sous forme d'une suite d'éléments de séquence. Les noms associés aux éléments de structure permettent de spécifier les contraintes de corrélation proposées par Eidhammer. La différence essentielle entre l'approche purement CSP et les autres outils réside ensuite dans la manière de générer les domaines (*a priori* une fois pour toutes dans le cadre CSP et dans Palingol), à la demande dans les autres outils existants.

Les variables " Éléments de séquence"

Une variable de ce type représente une unité élémentaire " élément de séquence " du motif recherché. Nous entendons par " élément de séquence " une chaîne de caractères ou mot.

Les variables " Éléments de structure"

Une variable de ce type représente une unité élémentaire " élément de structure " du motif recherché. Nous entendons par " élément de structure " une paire ou bien un triplet (voire plus) de chaînes de caractères partageant des interactions. Une unité de structure secondaire est par exemple composée de 2 chaînes de caractères dont l'une est le reverse complément de l'autre. Une unité de structure tertiaire de type pseudo-noeud peut être assimilée à un élément de structure secondaire. Une unité de structure tertiaire telle que la triple hélice est composée de 3 chaînes de caractères partageant des interactions.

On pourrait aussi envisager l'interaction d'un élément de séquence appartenant à un motif recherché (motif " maître ") avec un élément de séquence ou de structure d'un autre motif (motif " cible "). Ce dernier cas permettrait de traiter la recherche de snoARN simultanément avec la recherche de ses cibles potentielles.

Les variables définies par l'ensemble des outils généralistes existants sont obligatoires. Il n'est actuellement pas possible de définir de manière formelle des variables optionnelles. Il est cependant possible à partir d'outils tels que Palingol ou bien RnaMotif, qui possèdent un langage suffisamment puissant, de définir des variables optionnelles ou utilitaires.

La notion de variable alternative n'existe pas non plus dans les outils actuels. Il serait pourtant intéressant de conditionner l'existence d'une variable obligatoire à la présence d'une autre (les deux variables sont mutuellement exclusives). Enfin, la définition de la recherche simultanée de plusieurs motifs (dont l'un serait par exemple la cible de l'autre) n'est pas possible non plus.

Les domaines et les valeurs

La génération des valeurs des domaines *a priori* est un handicap certain (cas de Palingol). Il faut donc suivre l'idée contenue dans la plupart des outils, d'une génération des valeurs des domaines à la volée.

La recherche de solutions

La recherche de solutions doit pouvoir être abordée sous plusieurs formes. Du point de vue de l'ordre de la recherche d'une part, du point de vue de la génération de solutions d'autre part. L'ordre de la recherche doit pouvoir être réalisée sur la base de " l'unité la plus contrainte d'abord ". Une représentation du problème dans un cadre CSP devrait pouvoir fournir des pistes. Les solutions recouvrantes pourraient être gérées comme des ensembles de solutions. Il faudrait alors savoir caractériser ce que l'on appelle ensemble de solutions, produire cet ensemble de solutions sans en énumérer le contenu (trouver une représentation de cet ensemble, évaluer le nombre de solutions dans cet ensemble). Cela nous éviterait une énumération inutile de solutions. Il faudrait aussi proposer un moyen d'interroger cet ensemble de solutions et de classer ces solutions (notion de coût à exploiter).

3.9.1.2 Les contraintes

Les contraintes définies dans les outils généralistes sont soit :

- des contraintes unaires et sont alors utilisées pour la génération des domaines,
- soit des contraintes k-aires, encore appelées globales dans certains outils. Elles sont vérifiées en fin de branche d'un arbre de recherche.

Il serait judicieux, sachant qu'un grand nombre des contraintes sont des contraintes binaires, d'introduire cette classe de contraintes afin de réduire efficacement l'espace de recherche. En outre, l'introduction de contraintes permettant de cibler des interactions entre variables appartenant à des molécules différentes présente un attrait certain.

Le chapitre qui suit est dédié à l'étude de la modélisation des séquences nucléiques. Les choix faits pour représenter cette structure sont déterminants pour appréhender chacun des points mentionnés ci-dessus .

Chapitre 4

Modélisation du problème de la recherche de motif

Dans le chapitre précédent, nous avons présenté un état de l'art des différents logiciels généralistes pour la recherche de motifs structurés. Nous avons choisi d'effectuer cette comparaison en analysant les logiciels dans un cadre théorique commun : le cadre CSP. Cette analyse dans un cadre commun nous a permis de dégager les principales caractéristiques des logiciels généralistes de recherche de motifs (par exemple concernant la spécification des interactions entre molécules).

Notre objectif n'est pas simplement d'étudier les logiciels de recherche de motifs existant dans le cadre des CSP mais également de proposer une nouvelle méthode de recherche. Cette nouvelle méthode, basée elle aussi sur le cadre CSP, sera décrite dans le chapitre suivant. Mais avant de l'exposer, nous décrivons dans ce chapitre le cadre CSP, ainsi qu'une autre approche également utilisée pour la reconnaissance de motifs, l'analyse lexicale.

Ces deux approches sont basées sur des représentations différentes d'un motif structuré :

- Pour l'analyse lexicale, sur une représentation concise d'un ensemble de mots ou séquences au moyen d'un ensemble de règles courtes, appelées grammaires.
- Pour le cadre CSP, sur une représentation de type objet décrivant un ensemble d'éléments simples, connectés entre eux par un ensemble de dépendances pouvant être complexes.

L'analyse lexicale, ou processus de reconnaissance d'un mot, appliquée à l'étude des séquences biologiques comprend deux phases. Tout d'abord il faut, à partir d'une séquence ou d'un ensemble de séquences, construire une grammaire permettant de la (les) générer, via les règles de productions. Ensuite, cette grammaire est traduite en un automate capable de déterminer l'appartenance d'un motif quelconque au langage générée par la grammaire (donc aux séquences considérées). Le type de requête possible sur les séquences dépend du pouvoir descriptif de la classe du langage choisi : on peut représenter grâce à des grammaires, des palindromes, des sites spécifique etc...

Le cadre CSP, que nous avons brièvement décrit dans le chapitre précédent, permet une expression beaucoup plus souple des connaissances biologiques sur les séquences ou motifs, via des contraintes. La recherche de motif est prise en charge par un résolveur de contraintes, totalement indifférent au mode d'expression des contraintes. Ce formalisme offre un cadre naturel comme nous l'avons vu dans le chapitre III pour représenter la recherche d'un motif structuré

sous forme d'un réseau de contraintes.

L'objectif de ce chapitre est d'introduire de manière très générale les notions associées à la théorie des langages formels et aux problèmes de satisfaction de contraintes. Dans la conclusion de ce chapitre, nous donnerons les éléments qui nous ont conduit à privilégier le cadre CSP dans notre étude.

4.1 Approche par les grammaires

Un ensemble de **mots**, composant **un langage**, est représenté de manière concise par **une grammaire**. On appelle grammaire générative les règles de transformation qui permettent de générer l'ensemble des mots appartenant à ce langage. On distingue différents niveaux d'expressivité des grammaires selon une hiérarchie proposée par N. Chomsky [Cho57].

4.1.1 Définition d'une grammaire

Une grammaire est une liste de symboles et de règles de transformation où les symboles sont de deux types : (i) terminaux (les symboles visibles dans les mots) et (ii) non terminaux (les symboles -intermédiaires- qui permettent l'application des règles de transformation).

On définit une grammaire générative ou grammaire G par un quadruplet $< \sum, N, S, P >$ où :

$\sum$ est l'ensemble fini de symboles terminaux ou alphabet ou vocabulaire terminal.

N est un ensemble fini de symboles non terminaux.

S appelé symbole initial, appartient à N.

P est l'ensemble fini des productions de la forme $\alpha \rightarrow \beta$. Une telle règle est encore appelée règle de réécriture du fait qu'elle précise que la séquence de symboles α peut être remplacée par la séquence de symboles β.

Par convention, les majuscules (S) sont des symboles non terminaux et les minuscules (a, u, g, c) des symboles terminaux.

Un mot (séquence de symboles de $\sum$) est généré par la grammaire G si et seulement si il peut être obtenu en partant du symbole initial et en appliquant des règles de productions de G jusqu'à ce que la chaîne résultante obtenue par des réécritures successives ne contienne que des symboles terminaux.

Si $\alpha \rightarrow \beta$ est une règle de production d'une grammaire, on dit que α est la partie gauche et β la partie droite de la production. L'opération autorisée dans les grammaires présentées ici est la réécriture d'une chaîne de symboles en symboles terminaux par application d'une règle de production.

Exemple :

- Soit l'ensemble des symboles : $\sum = \{a, g, S\}$, où a, g sont les symboles terminaux.
- Soit les règle de transformation $\{S \rightarrow aS$, $S \rightarrow gS$, $S \rightarrow a$ ou $S \rightarrow g\}$, permettant quatre possibilités de transformer S.

A chaque étape de production d'un mot, le symbole S est remplacé par l'une des trois possibilités. Par exemple, le mot $aggg$ (appartenant au langage L) peut être produit par la séquence de transformations suivantes :

$S \rightarrow aS \rightarrow agS \rightarrow aggS \rightarrow aggg$

4.1.2 Hiérarchie de Noam Chomsky

La classification proposée par Chomsky [Cho57] permet de regrouper les grammaires en fonction de leurs formes étroitement liées à leur expressivité. On distingue quatre catégories selon la forme de leurs règles, énumérées selon un gradient croissant d'expressivité (l'ensemble des grammaire d'un niveau donné sont incluses dans l'ensemble des grammaires de niveaux supérieurs) : grammaires régulières $\subset$ grammaires hors-contextes $\subset$ grammaires sensibles au contexte $\subset$ grammaires non restreintes.

Grammaire régulière ou rationnelle : Les règles sont de la forme : $S \rightarrow aS$ ou $S \rightarrow a$. La partie gauche est un symbole non terminal et la partie droite contient un symbole terminal et/ou non terminal. Par exemple:

$$\left\| \begin{array}{l} \sum = \{a, g, S\} \\ L = \{a, g, aa, ag, ga, gg, aaa, ...\} \\ S \rightarrow aS|gS|a|g \end{array} \right.$$

Grammaire hors contexte ou algébrique : Les règles sont de la forme : $S \rightarrow \beta$ où β est une chaîne quelconque contenant des symboles terminaux ou non. S se ré-écrit en β quel que soit son contexte. La partie gauche est exclusivement un symbole non terminal. Par exemple :

$$\left\| \begin{array}{l} \sum = \{a, g, S\} \\ L = \{aa, gg, agga, gaag, gggg, aaaa, ...\} \\ S \rightarrow aSa|gSg|aa|gg \end{array} \right.$$

Grammaire sensible au contexte ou contextuelle : Les règles ne s'appliquent que si la partie gauche se trouve dans un contexte donné. Elles sont de la forme : $aSc \rightarrow \beta$. S peut se ré-écrire en β si et seulement s'il se trouve dans un contexte où a le précède et c le suit. Les parties gauches/droites sont dans un contexte constitué de chaînes de caractères terminaux et/ou non terminaux. Par exemple :

$$\left\| \begin{array}{l} \sum = \{a, g, S_1, S_2\} \\ L = \{(ag)^n | n \geq 1\} \\ S_1 \rightarrow aS_2 \\ aS_2 \rightarrow agS_1 \\ S_2 \rightarrow g \end{array} \right.$$

Grammaire générale : C'est une forme générale où les parties gauche et droite sont des chaînes de caractères sans restriction, permettant toutes les combinaisons de symboles. Les règles sont de la forme : $A \rightarrow B$. Toute occurrence de la chaîne A peut se ré-écrire en B.

4.1.3 Langages et analyse lexicale

N. Chomsky donne une définition des langages à partir des grammaires génératives [Cho57] :

Définition 4 (Un langage) *:*
Un langage $L(G)$ est produit par une grammaire G qui capture l'ensemble des caractéristiques communes à toutes les chaînes de caractères appartenant à ce langage.

La classification proposée par Chomsky [Cho57] permet de regrouper les langages en fonction de la forme de leur grammaire et par conséquent de leur expressivité :

- Langages réguliers ou rationnels (description de structures linéaires).
- Langages hors-contexte ou algébriques (description de structures récursives, hiérarchiques, telles que les palindromes).
- Langages contextuels ou sensibles au contexte (description de répétitions).
- Langages non restreints ou récursivement énumérables (tout).

Un certain nombre d'opérations s'appliquent aux langages parmi lesquels, dans le cadre de l'analyse lexicale, l'union, la concaténation ou la fermeture de Kleene .

Définition 5 (L'union, la concaténation ou la fermeture de Kleene) *:*

L'union de deux langages L_1 et L_2 est le langage défini sur $\sum_1 \cup \sum_2$ contenant tous les mots qui sont soit contenus dans L_1, soit contenus dans L_2.
La concaténation (ou produit) de L_1 et L_2 est le langage défini sur $\sum_1 \cup \sum_2$ formé d'un mot de L_1 suivi d'un mot de L_2.
La fermeture itérative de L_1 (ou fermeture de Kleene ou itérée de L_1) est l'ensemble des mots formés par une concaténation finie de mots de L_1.

A partir de ces opérations, un langage permet de définir ainsi un ensemble de mots, unités linguistiques formées selon des règles précises à l'aide d'un alphabet. Les mots appartenant a ce langage sont reconnus par un analyseur lexical associé à la grammaire.

4.1.4 Analyse lexicale par automate

Les grammaires sont décrites comme des modèles de production de chaînes de caractères. Les automates, quant à eux, sont décrits comme des analyseurs syntaxiques qui acceptent ou rejettent une chaîne de caractères donnée : une chaîne est acceptée si et seulement si elle peut être produite par la grammaire.

4.1.4.1 Définition de l'automate

Un automate se définit par un ensemble d'états E, où l'on distingue un état initial e_0 et un ou plusieurs états finaux, un vocabulaire X (ensemble de symboles) et une fonction de transition associant à certains couples de (E, X) un ou plusieurs éléments de E. Par exemple, la transition $(e_i, x) \rightarrow e_j$ signifie : quand on est à l'état e_i et qu'on lit le symbole x, on passe à l'état e_j. Un

$$\begin{cases} \text{Expression régulière :} & (\ abb\ )^*\ abb \\ \text{Grammaire :} & S \rightarrow aS \mid bS \mid abb \\ \text{Langage :} & L = \{abb, aabb, babb, aaabb, ababb, baabb, bbabb, ...\} \end{cases}$$

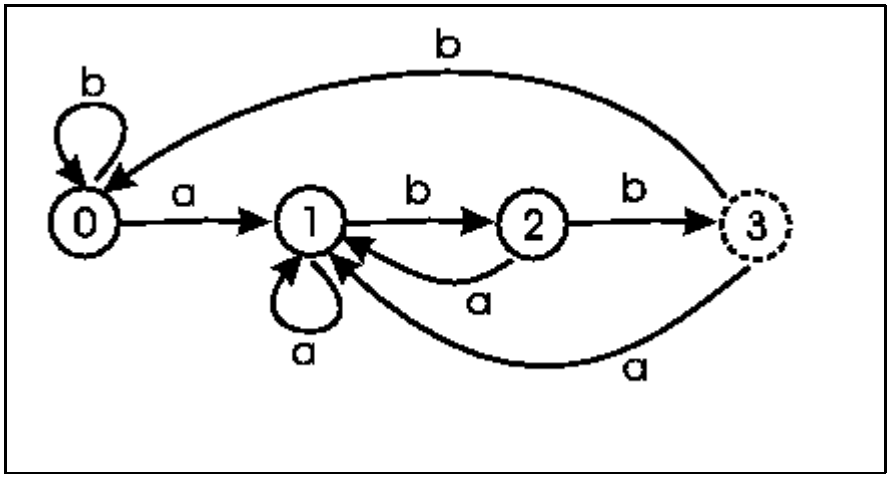


FIG. 4.1 – *Automate de reconnaissance de l'expression régulière :* $(a|b)^*abb$*. État initial :* 0 *et état final :* 3.

automate est déterministe si pour tout état e_i et pour tout symbole x, il existe au plus un état e tel que $(e_i, x) \rightarrow e$.

Un mot est reconnu par un automate (Fig 4.1) si et seulement s'il est l'étiquette d'un chemin (pouvant parcourir plusieurs fois les mêmes noeuds) qui part de l'état initial et arrive à un état terminal. On lit le mot symbole après symbole, tout en se déplaçant dans l'automate en suivant les transitions étiquetées par le symbole courant. Si après avoir lu tout le mot, on se trouve dans un état terminal de l'automate, le mot est reconnu ; sinon le mot n'est pas reconnu. On définit ainsi le langage reconnu par un automate, noté $L(A)$, comme l'ensemble des mots reconnus. Un analyseur syntaxique pour un langage est un programme qui prend en entrée une chaîne de caractères M et répond "oui" si M est un mot du langage et "non" autrement.

4.1.4.2 Correspondance langages-automates

Selon la complexité d'un langage, en d'autres termes sa classe, on peut associer un automate permettant l'analyse lexicale des mots appartenant au langage considéré [Sea97].

- langages réguliers : automates à état fini (Fig 4.1).
- langages hors-contextes : automates à pile.
- langages contextuels : automates à mémoire bornée.
- langages non restreint: automates de Turing.

Les grammaires régulières et hors-contextes possèdent des analyseurs lexicaux optimaux basés sur des algorithmes de complexité polynomiale. L'analyse des langages contextuels est NP-complet et celle des langages non restreints est indécidable (dans le cas général, l'analyse lexicale d'un mot n'appartenant pas à un langage non restreint peut ne pas se finir).

4.1.5 Application des grammaires aux séquences nucléiques

Les biologistes utilisent depuis longtemps des techniques issues de la linguistique pour décrire les processus cellulaires impliquant les séquences biologiques. Dans le cas particulier de la modélisation d'acides nucléiques, une famille de séquences est alors caractérisée par la présence ou l'absence de motifs particuliers. En adoptant l'approche "linguistique", un ensemble de séquences peut plus généralement être représenté par des règles grammaticales. On se place ainsi dans la problématique de l'analyse grammaticale qui étudie ce problème aussi bien sur le plan théorique qu'algorithmique (Tab 4.1).

Langage	**Automate**	**Complexité de la Reconnaissance**	**Dépendance**	**Problème en Biologie**
Langage régulier	Automates à états finis	Linéaire	locale 	Reconnaissance de la structure primaire
Langage hors contexte	Automates à piles	Polynomial $O(n^3)$ en général	voisine 	Reconnaissance de la structure secondaire
Langage sensible au contexte	Automates à mémoire bornée	NP-complet	croisée 	Reconnaissance de la structure secondaire avec pseudo-noeud
Langage récursivement énumérable	Machine de Turing	Indécidable		Reconnaissance maximale

TAB. 4.1 – *Hiérarchie des grammaires de Noam Chomsky, d'après [Sea02, Del03].*

Searls s'est intéressé au problème de la modélisation de la structure secondaire/tertiaire des ARN. Il a démontré que le niveau des langages hors contexte capture les structures telles que les palindromes (structure secondaire des ARN) [SD93, Sea99] mais que les répétitions appartiennent au niveau suivant. Il a cherché une approche minimaliste pour trouver une grammaire pouvant prendre en compte les copies sans aller jusqu'au niveau des langages contextuels. Les grammaires indexées, dont le langage est strictement entre le niveau hors contexte et contextuel permettent de prendre en compte les copies. Les langages indexés sont similaires aux langages hors contextes en termes de propriétés de décidabilité et l'analyse lexicale des grammaires associées est NP-complet.

Searls a utilisé le formalisme `DCG` (*Definite Clause Grammars*) pour implémenter de telles grammaires dans le logiciel GENLANG [DS94]. Les grammaires `DCG` sont rattachées au langage de programmation logique Prolog qui constitue un analyseur pour ces grammaires. GenLang a été utilisé, en particulier, pour la représentation et la prédiction des gènes d'ARNt dans une banque de données. En utilisant son logiciel GenLang pour la recherche des séquences d'ARNt dans plusieurs séquences biologiques, il obtient des résultats comparables à ceux obtenus par tRNAscan.

E. Rivas et S.R. Eddy [RE00a] ont proposé une grammaire hors-contexte pour les structures de l'ARN de type pseudo-noeuds. Cette grammaire utilise des symboles auxiliaires et des règles de réarrangement qui la différencient d'une grammaire hors-contexte conventionnelle. L'algorithme de recherche a une complexité plus élevée qu'un analyseur lexical hors-contexte mais reste polynomiale.

Un formalisme probabiliste a été utilisé pour la modélisation des structures secondaires d'ARN. Il est basé sur les grammaires stochastiques hors contexte. Une grammaire stochastique hors contexte est une grammaire hors contexte pour laquelle une probabilité est affectée à chacune de ses productions. Cela revient à affecter une probabilité à chaque séquence dérivée. L'intérêt d'une telle représentation est de pouvoir choisir parmi toutes les structures secondaires trouvées celle qui a la probabilité la plus élevée. Ce type de grammaire a aussi été largement utilisée pour modéliser les familles d'ARN [SBH+94, ED94].

4.2 Approche par les CSP

Dans le domaine des CSP, la notion de contrainte désigne un énoncé déclaratif mettant en relation les données d'un problème. Une contrainte constitue donc un élément de représentation de la connaissance relative à un problème. Un résolveur CSP permet, d'une part, d'exprimer dans un formalisme déclaratif les contraintes qui décrivent un problème, et fournit, d'autre part, des algorithmes capables de déterminer sa cohérence et de le résoudre s'il admet une solution.

L'approche fondée sur les contraintes permet ainsi de séparer l'expression du problème (aspect représentation) et les techniques algorithmiques de résolution (gestion des contraintes).

4.2.1 Définition d'un système de contraintes

Un problème de satisfaction de contraintes est défini par un ensemble de variables dont les valeurs sont restreintes par des contraintes. Le CSP à considérer est un triplet $P = (X, D, C)$ où :

- $X = \{x_1, ...x_n\}$ est un ensemble de n variables,
- $D = \{d_1, ..., d_n\}$ représente les domaines de valeurs de ces variables (ensemble de valeurs pouvant être prises par les variables),
- $C = \{c_1, c_2, ...c_m\}$ est un ensemble de m contraintes.
  - $variables(c_j) = \{x_{j_1}, ...x_{j_{p_j}}\}$ est l'ensemble des p_j variables liées par c_j ; p_j est appelé arité de la contrainte c_j.
  - $relation(c_j)$ est la relation associée à la contrainte c_j sur le produit cartésien de domaines : $d_{j_1} \times ... \times d_{j_{p_j}}$. (Le domaine d_{j_1} est associé à la variable x_{j_1} et le domaine $d_{j_{p_j}}$ à la variable $x_{j_{p_j}}$), c'est à dire l'ensemble des p_j-uplets $(x_{j_1}, ..., x_{j_{p_j}})$ admis par la contrainte. Elle liste les combinaisons de valeurs autorisées pour la variable.

On distingue deux types de contraintes :

Contrainte donnée en intension : La contrainte est donnée sous forme d'un prédicat sur les variables.

Contrainte donnée en extension : La contrainte s'exprime sous la forme d'un ensemble de n-uplets de valeurs permises pour les variables sur lesquelles elle porte.

Dans la suite de cette section, on considérera exclusivement les contraintes binaires (contraintes avec deux variables).

L'exemple (Fig 4.2) définit un CSP composé de trois variables où le domaine de chacune est composé de 2 ou 3 valeurs. Les contraintes spécifiées sont binaires et sont données en intention et en extension.

$$\begin{cases} V = \{x_1, x_2, x_3\} \\ D = \begin{cases} D_1 = \{2,3,4\} \\ D_2 = \{1,2,3\} \\ D_3 = \{2,3\} \end{cases} \\ C = \begin{cases} \text{en intension} & \text{en extension} \\ C_1 : x_1 > x_2 & C_1(x_1,x_2) = \{(2,1),(3,1),(3,2),(4,1),(4,2),(4,3)\} \\ C_2 : x_2 < x_3 & C_2(x_2,x_3) = \{(1,2),(1,3),(2,3)\} \\ C_3 : x_1 < x_3 & C_3(x_1,x_3) = \{(2,3)\} \end{cases} \end{cases}$$

FIG. 4.2 – *Formalisation du problème avec l'ensemble V de variables, l'ensemble D des domaines de valeurs des variables et l'ensemble C des contraintes posées sur les variables de l'ensemble V.*

Définition 6 (Affectation, instanciation, solution et cohérence) *:*
Le processus de donner une valeur à une variable est appelé affectation, formant un couple (x_i, v_i).

Un ensemble d'affectations $(x_i, v_i)_{i \in I \subseteq 1...n}$ est une instanciation partielle. C'est une instanciation globale si $I = 1...n$.

Une instanciation $(x_i, v_i)_{i \in I}$ est cohérente si elle satisfait toutes les contraintes c_j telles que l'ensemble des variables de c_i est affectée dans l'instanciation et que les combinaisons de valeurs prises sont autorisées.

Une solution d'un CSP est une instanciation globale cohérente.

En général, l'établissement de l'existence d'une solution d'un CSP est un problème NP-complet. La complexité des algorithmes, permettant de le résoudre dans le cas général, croît exponentiellement avec la taille du problème, dans le pire des cas, laquelle est liée aux nombres de variables, de contraintes et à la taille des domaines de valeurs des variables.

Un CSP binaire (dont toutes les contraintes sont d'arité ≤ 2) est modélisable par un graphe (Fig 4.3) où les sommets correspondent à des variables et les arêtes représentent les contraintes binaires reliant deux variables.

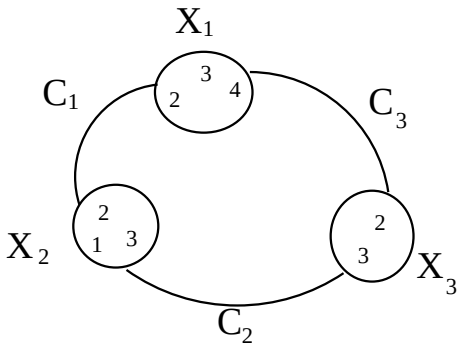


FIG. 4.3 – *Graphe du CSP de la figure 4.2.*

4.2.2 Techniques de résolution d'un CSP

La résolution d'un CSP peut avoir les objectifs suivants :
- déterminer s'il existe ou non une solution.
- exhiber une solution (objectif le plus courant).
- exhiber plusieurs ou toutes les solutions.

4.2.2.1 La méthode générer/tester

La méthode `générer/tester` (Fig 4.4) est une méthode naïve et peu efficace pour résoudre un CSP. Elle énumère les instanciations possibles et vérifie leur cohérence. Si l'on considère, par exemple, qu'un CSP est composé de n variables, de m contraintes, et que chaque domaine comporte d valeurs, le nombre d'instanciations à tester pour démontrer l'incohérence de ce problème suivant cette méthode est au maximum de d^n. On peut
avoir à évaluer un maximum de $m \times d^n$ contraintes.

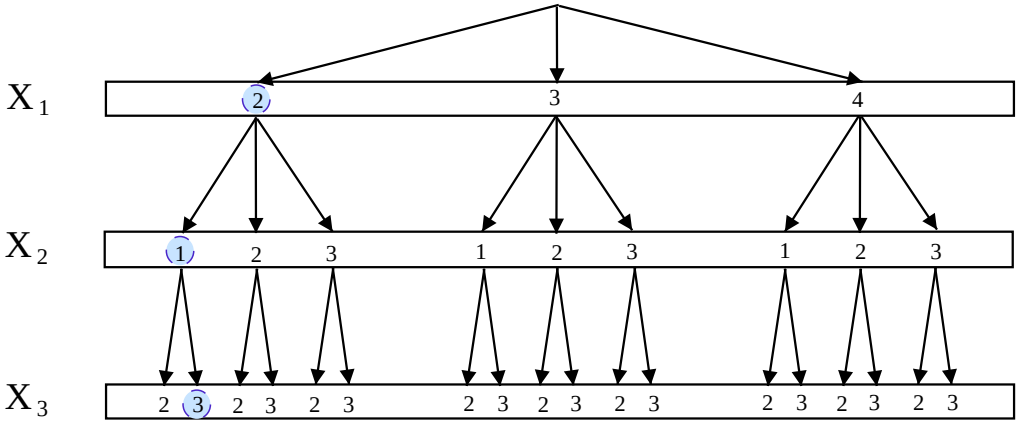


FIG. 4.4 – *Enumération de l'ensemble des configurations possibles sous la forme d'un arbre de recherche. Les valeurs entourées d'un cercle correspondent à une instanciation cohérente du problème.*

En pratique, cette méthode permet de générer un arbre complet de profondeur n de facteur de branchements d à partir d'un ordre fixe sur les variables. Chaque branche (de longueur n) reliant la racine à une feuille de l'arbre représente une instanciation complète. Les feuilles (il y en a d^n) sont étiquetées cohérentes / incohérentes selon la cohérence de l'instanciation complète. La génération d'un tel arbre peut prendre un temps prohibitif, d'autant que seule la génération des instanciations cohérentes nous intéresse. La méthode suivante permet de réduire la taille de l'arbre généré en vérifiant au fur et à mesure de leur génération la cohérence des instanciations partielles.

4.2.2.2 La méthode `retour-arrière`

Le principe `retour-arrière` (*backtrack*) en profondeur d'abord (Fig 4.5) , plus efficace que le précédent, instancie séquentiellement les variables. À chaque fois qu'une nouvelle variable est instanciée, on teste si l'instance partielle générée est localement cohérente. Si une

incohérence est détectée, un retour-arrière est effectué sur la dernière variable instanciée et une nouvelle valeur de son domaine est testée.

```
Algorithme : retour-arrière
Données : X_NI : variables non instanciées
          I : instanciation courante
Résultat : [I] : instanciations cohérentes des variables.
si X_NI = ∅ alors
    /* On a donné une valeur à toutes les variables avec
    succès */
    retourner I
sinon
    /*Nouvelle variable à instancier*/
    Choisir x_i ∈ X_NI;
    pour chaque valeur v_r de x_i faire
        si I ∪ {x_i = v_r} est cohérent alors
            retour-arrière (X_NI − {x_i} , I ∪ {x_i = v_r}) ;
```

FIG. 4.5 – *Procédure de* `retour-arrière`.

Cet algorithme correspond à un parcours en profondeur d'abord de l'arbre de recherche. Il permet d'éliminer de la recherche le sous espace des valeurs de toutes les variables (suivant la dernière variable instanciée) à chaque fois qu'une instanciation partielle viole une des contraintes du problème.

La figure 4.6 représente l'élagage réalisé par le `retour-arrière`. Lorsque l'instanciation partielle $(x_1 = 2, x_2 = 2)$ est réalisée, l'algorithme détecte la violation de la contrainte c_1 et évite d'explorer les deux instanciations complètes qui l'étendent. On voit clairement, en comparaison de la méthode `générer/tester`, que 6 configurations ont été éliminées. L'arbre de recherche ne contient plus que 24 noeuds et 12 configurations alors qu'il comporte 30 noeuds et 18 configurations sur la figure 4.4.

Le `retour-arrière` gagne en efficacité par rapport à la méthode `générer/tester`, mais il peut encore être amélioré. En effet, l'algorithme `retour-arrière` effectue un parcours aveugle de l'espace de recherche, en profondeur d'abord, sans prendre en compte des informations évidentes sur l'incohérence d'instanciations partielles et re-découvre, de manière répétée, les mêmes incohérences locales. Par exemple, la figure 4.6 montre que l'instanciation partielle $(x_3 = 2)$ est explorée 3 fois alors que toutes les valeurs autorisées de x_1 induisent une violation de c_3 par $(x_1, x_3 = 2)$.

Plusieurs types de techniques permettent d'augmenter l'efficacité de résolution des CSP :

– les techniques de filtrage ;

– les techniques de `retour-arrière` intelligent ;

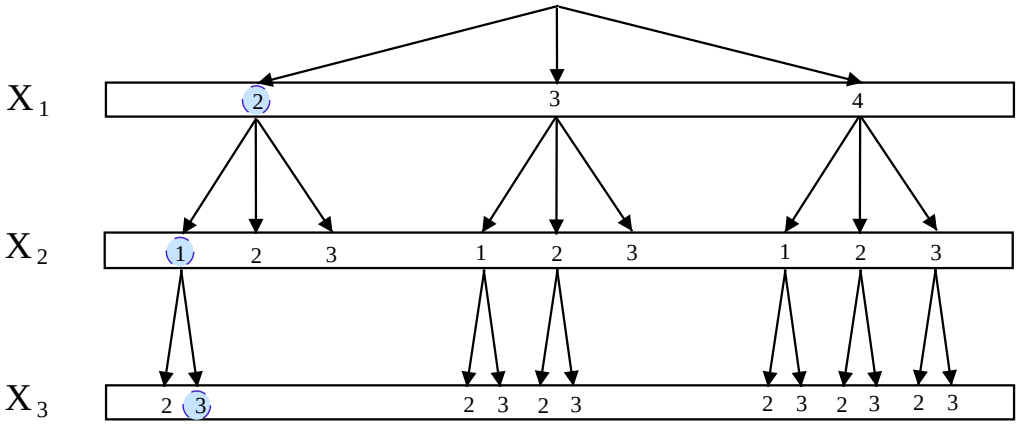


FIG. 4.6 – *Ensemble des configurations énumérées par la méthode de* `retour-arrière`.

– les heuristiques de guidage : l'ordonnancement des variables statique ou dynamique.

4.2.2.3 Techniques de filtrage

L'objectif d'un filtrage est de produire à partir d'un CSP initial un CSP équivalent (même ensemble de solutions) mais plus simple potentiellement. Techniquement, les algorithmes de filtrage peuvent améliorer l'algorithme de `retour-arrière` en supprimant dans les domaines des variables non affectées les valeurs qui sont incompatibles avec l'affectation partielle courante (ne peuvent mener à une solution).

Définition 7 (Cohérence de noeud d'un CSP) *:*
Un CSP est noeud-cohérent si pour toute variable x_i de X, et pour toute valeur v de $D(x_i)$, l'affectation partielle (x_i, v) satisfait toutes les contraintes unaires.

La cohérence de noeud restreint les domaines de variables en testant les contraintes unaires. Cette restriction peut être rendue plus efficace en testant les variables par paires : c'est le filtrage par arc-cohérence.

Définition 8 (Arc-cohérence d'un CSP) *:*
Un CSP est arc-cohérent si pour tout couple de variables (x_i, x_j) de X reliés par une contrainte $c_{i,j}$, et pour toute valeur v_i appartenant à $D(x_i)$, il existe une valeur v_j appartenant à $D(x_j)$ appelée support de (x_i, v_i) sur x_j telle que l'affectation partielle $(x_i, v_i), (x_j, v_j)$ soit cohérente.

On peut calculer un CSP équivalent (mêmes solutions) et arc-cohérent en supprimant les valeurs de variables ne pouvant être étendues à une seconde variable. L'arc-cohérence n'implique pas l'existence d'une solution mais l'existence d'un domaine vide dans un CSP arc-cohérent équivalent est une preuve d'absence de solution.

Le principe de l'algorithme de filtrage par propagation de contraintes, illustré par l'exemple de la figure 4.7, est de supprimer de chaque domaine toutes les valeurs sans support sur une autre variable (une valeur v_a d'un domaine d_i a un support v_b dans le domaine d_j si le couple (v_a, v_b) est autorisé par la contrainte $c_{i,j}$7liant les variables x_i et x_j).

$$\begin{cases} V = \{x_1, x_2, x_3\} \\ D = \begin{cases} D_1 = \{2,3,4\} \\ D_2 = \{1,2,3\} \\ D_3 = \{2,3\} \end{cases} \\ C = \begin{cases} C_1 \ : \ x_1 > x_2 \\ C_2 \ : \ x_2 < x_3 \\ C_3 \ : \ x_1 < x_3 \end{cases} \end{cases}$$

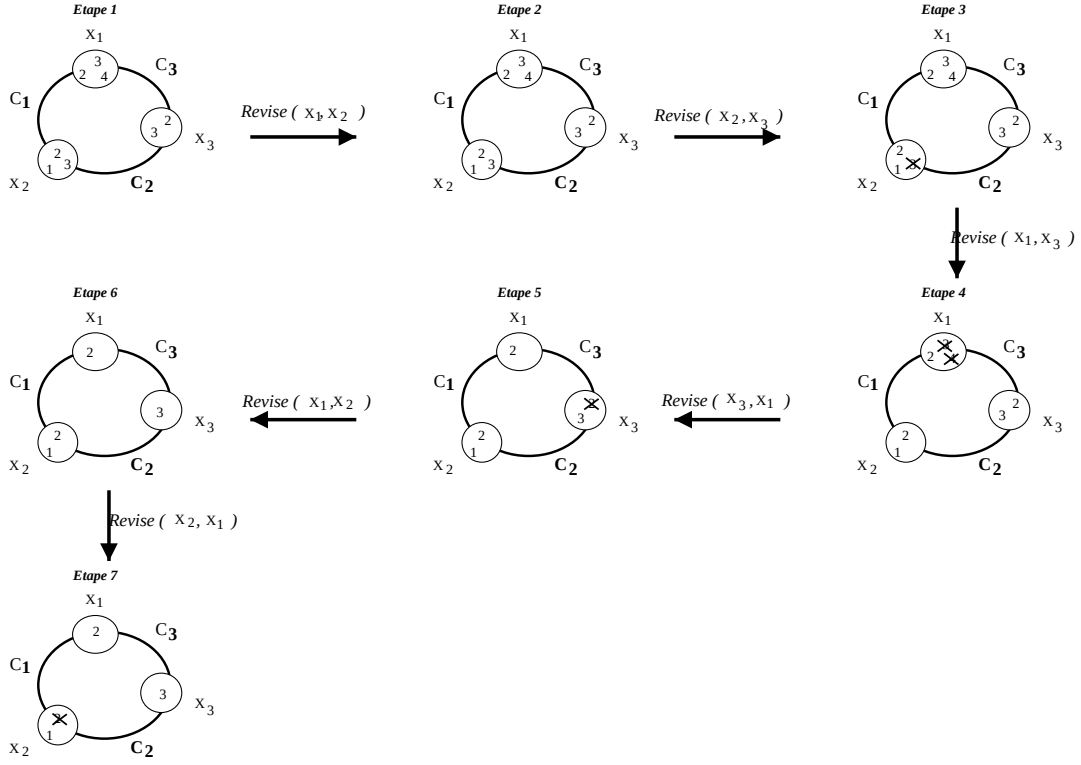


FIG. 4.7 – *Application d'un algorithme de cohérence d'arc sur le CSP.*

Cette suppression est effectuée par la fonction `Revise`(x_i, x_j) qui supprime du domaine de x_i toutes les valeurs n'ayant pas de support dans le domaine de x_j. Cette fonction retourne une valeur booléenne indiquant si au moins une suppression a effectivement eu lieu.

On dit que la fonction `Revise`(x_i, x_j) rend le couple (x_i, x_j) arc-cohérent (mais pas le couple (x_j, x_i)).

Algorithme : Revise

Données : x_i, x_j
SUPPRESSION ← Faux;
pour chaque *valeur* v_a *de* d_i **faire**
  si *il n'existe pas de valeur* v_b *dans* d_j *support de* v_a
  alors
    supprimer v_a de d_i;
    SUPPRESSION ← Vrai;
retourner SUPPRESSION

FIG. 4.8 – *Fonction* Revise. v_a *est une valeur du domaine* d_i *de* x_i *et* v_b *est une valeur de* d_j *de* x_j.

La fonction Revise (Fig 4.8) permet d'assurer l'arc-cohérence d'un CSP de la manière suivante. Elle peut être appliquée à tous les couples de variables. La modification du domaine de la variable x_i entraînant l'apparition de nouvelles valeurs arc-incohérentes sur d'autres variables reliées par une contrainte à x_i, plusieurs "passes", ou applications successives de Revise peuvent être nécessaires avant que la cohérence d'arc soit assurée. C'est le principe de l'algorithme AC-1 (Fig 4.9).

Algorithme : AC-1

Données : $Q \leftarrow \{(x_i, x_j)/(x_i, x_j)$ reliées par une contrainte. $\}$
répéter
  CHANGE ← Faux;
  pour chaque (x_i, x_j) *de* Q ;
  faire
    CHANGE ← Revise(x_i, x_j) OU CHANGE;
jusqu'à *NON CHANGE*;

FIG. 4.9 – *Algorithme* AC-1. Q *: ensemble des paires de variables reliées par une contrainte. Le booléen retourné par la fonction indique si le domaine de* x_i *a été ou non modifié.*

L'algorithme AC-1 (Fig 4.9) permet d'assurer la cohérence d'arc d'un CSP. Dans cet algorithme, à chaque fois que la fonction Revise supprime des valeurs du domaine d'une variable donnée, elle est appelée à nouveau sur tous les arêtes du graphe jusqu'à ce que la fonction Revise retourne Faux pour tous les couples de variables. La complexité d'AC-1 est définie par le nombre d'appels à la fonction Revise fois son coût. Si l'on considère un CSP formé de n

variables, m contraintes et dont les domaines ont une taille d, on obtient une complexité globale pour AC-1 de $O(nmd^3)$.

L'algorithme AC-1 peut être amélioré. En effet, à chaque "passe" il examine tous les couples de variables, même si les variables impliquées dans ces couples ne sont liées à aucune variable dont le domaine a été modifié récemment par une appel à la fonction Revise.

Afin d'éviter de nombreux appels à la fonction Revise, on peut maintenir une liste explicite de tous les couples de variables à tester afin d'assurer l'arc-cohérence. C'est le principe de l'algorithme AC-3 (Fig 4.10).

Algorithme : AC-3

Données : $Q \leftarrow \{(x_i, x_j)/(x_i, x_j)$ reliées par une contrainte.$\}$
tant que Q *est non vide* **faire**
  extraire un arc (x_k, x_m) de Q;
  si *Revise*(x_k, x_m) **alors**
    $Q \leftarrow Q \cup \{(x_i, x_k)/(x_i, x_k)$ est un arc $\}$;

FIG. 4.10 – *Algorithme* AC-3.

L'amélioration apportée par AC-3 [Mac77] consiste à utiliser un ensemble Q pour stocker à chaque fois que le domaine de valeurs d'une variable a été modifié, tous les arcs ayant pour origine ce dernier noeud (toutes les contraintes posées sur cette variable). La complexité globale de AC-3 est de $O(md^3)$.

Un certain nombre d'algorithmes présentant une meilleure complexité théorique ont été par la suite proposés. Par exemple AC-4 [MH86] qui, en utilisant une structure de donnée sophistiquée, réduit la complexité théorique à $O(md^2)$.

4.2.2.4 Techniques d'amélioration du retour-arrière

Le filtrage par cohérence d'arc correspond à une étape préalable à la recherche de solutions : l'espace de recherche est élagué, puis la recherche est lancée. On peut étendre cette idée en entrelaçant filtrage et exploration : à chaque étape de la recherche, une nouvelle phase de filtrage est relancée. La propriété de cohérence d'arc n'est plus obtenue une seule fois préalablement à la recherche mais est, en quelque sorte, maintenue tout au long de cette recherche.

Pour mettre ce principe en oeuvre, on va ajouter, à chaque étape de la recherche, une étape de filtrage supplémentaire sur les domaines des variables non affectées en enlevant les valeurs "localement incohérentes". On peut effectuer les différents filtrages vus précédemment, correspondant à différents niveaux de cohérence locale et des réductions plus ou moins importantes sur les domaines des variable.

Si l'on considère que les n variables d'un CSP sont ordonnées, on peut définir des applications restreintes de la procédure Revise à partir d'une variable x_i donnée :

– backward checking : on applique la procédure Revise à tous les couples (x_i, x_k)

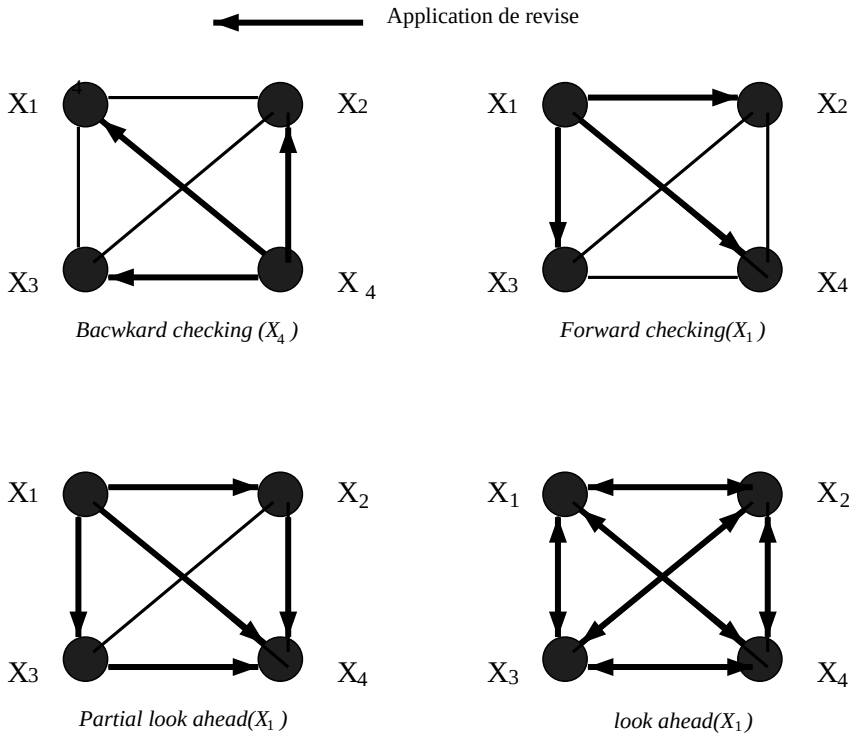


FIG. 4.11 – *Arc-cohérences dégradées (d'après [Sch93]).*

tels que $1 \leq k < i$;

- `forward checking` : on applique la procédure `Revise` à tous les couples (x_k, x_i) tels que $i < k \leq n$;
- `partial look ahead` : on applique la procédure `Revise` à tous les couples (x_j, x_k) tels que $i \leq j < k \leq n$;
- `full look ahead` : on applique la procédure `Revise` à tous les couples (x_j, x_k) tels que $i \leq j \neq k \leq n$;

Les principes d'anticipation de propagation de contraintes mis en oeuvre par ces différents algorithmes sont illustrés par la Figure 4.11. L'avantage que l'on peut tirer de la réduction de l'espace de recherche dépend considérablement des problèmes traités. Parmi les méthodes prospectives les plus connues, on peut citer le `forward checking` qui utilise une forme partielle de cohérence d'arc par rapport au voisinage immédiat de la variable affectée. Le `real full look ahead` ou algorithme de maintien d'arc cohérence `MAC` [SF94] maintient un CSP arc-cohérent à chaque noeud. Ce dernier est actuellement considéré comme étant le "meilleur" algorithme général pour résoudre les CSPs.

4.2.2.5 Techniques utilisées pour guider la recherche

Les algorithmes que nous venons d'étudier ne disent rien sur l'ordre dans lequel on doit instancier les variables, ni sur l'ordre dans lequel on doit affecter les valeurs aux variables. Ces deux ordres peuvent influencer considérablement l'efficacité de ces algorithmes.

On peut ainsi intégrer des heuristiques pour déterminer l'ordre dans lequel les variables et

les valeurs doivent être considérées.

Les heuristiques concernant l'ordre d'instanciation des valeurs sont généralement dépendantes de l'application considérée et difficilement généralisables. En revanche, il existe de nombreuses heuristiques d'ordre d'instanciation des variables qui permettent bien souvent d'accélérer considérablement la recherche. L'idée générale consiste à instancier en premier les variables les plus "critiques", c'est-à-dire celles qui interviennent dans beaucoup de contraintes et/ou qui ne peuvent prendre que très peu de valeurs. L'ordre d'instanciation des variables peut être :

statique : l'ordre d'instanciation est fixé avant de commencer la recherche (Fig 4.12). On remarque dans l'exemple de la figure 4.12 une diminution du nombre de noeuds et de feuilles à explorer (les valeurs des variables inspectées) apportée par l'ordre $X_3, X_2, X1$ par rapport à l'ordre $X_1, X_2, X3$.

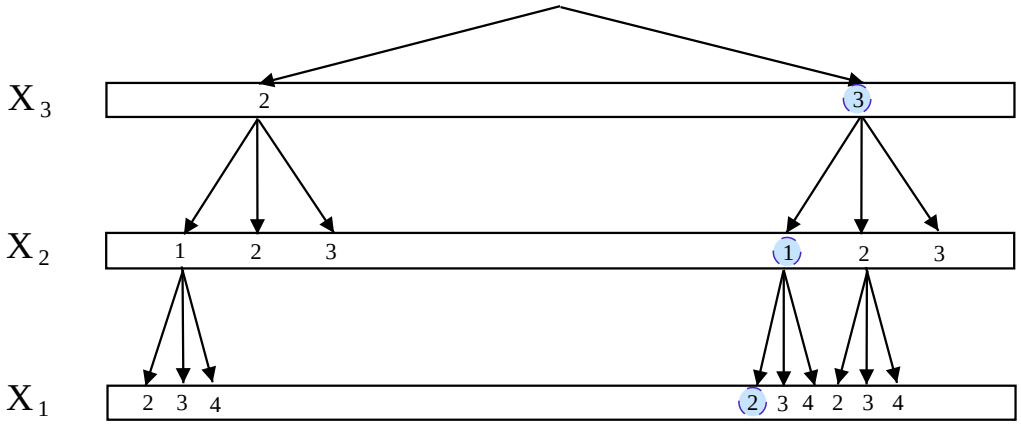


FIG. 4.12 – *Influence de l'ordre d'instanciation des variables sur l'application d'un algorithme de backward checking.*

dynamique : la prochaine variable à instancier est choisie dynamiquement à chaque étape de la recherche.

Par exemple, l'heuristique "échec d'abord" consiste à choisir, à chaque étape, la variable dont le domaine a le plus petit nombre de valeurs localement cohérentes avec l'affectation partielle en cours. Du fait de l'évolution de la taille des domaines au cours de la résolution du CSP, cet heuristique est surtout intéressant en conjonction avec la propagation des contraintes.

4.2.3 Application des CSP aux séquences nucléiques

Des systèmes utilisant le cadre CSP comme outil de représentation et de résolution tout au long du processus de détermination d'un modèle structural d'ARN ont été développés pour traiter les problèmes suivants : détermination de structures secondaires [GW94]; visualisation de structures secondaires [MGEW93]; détermination d'une structure tertiaire [AWN94, MTG+91, BK99].

Dans le cadre plus particulier de la recherche de motifs structurés, Eidhammer *et al.* [EGJR01] se sont intéressés aux formalismes à base de contraintes pour décrire et rechercher des motifs

structurés dans les séquences biologiques. Un motif est décrit, d'une part, au moyen d'une expression logique entre ses éléments, d'autre part, par l'ensemble des contraintes que doivent respecter ces éléments. Avec ce formalisme, il est possible de décrire des motifs de type copie et les palindromes. Les auteurs ont proposé deux approches pour effectuer la recherche.

La première approche (*CLP : constraint logic programming language*) développe un langage de contraintes à partir du langage de programmation logique `Prolog`. Satisfaire les contraintes revient à rechercher une valeur solution en élaguant les branches d'un arbre de recherche. L'utilisation des techniques de CSP comme deuxième approche permet de réduire l'arbre de recherche avant l'échec mais en induisant plus de travail à chaque noeud de l'arbre de recherche. Cette approche a donné lieu à une implémentation préliminaire par les mêmes auteurs et a fournit le point de départ à notre réflexion méthodologique.

Une approche intéressante de recherche de motifs basée sur les graphes a été proposée par S. Vialette [Via04]. L'auteur s'est intéressé dans le cadre de la prédiction de structure secondaire d'ARN aux problèmes algorithmiques sur des ensembles de 2-intervalles où les trois configurations proposées modélisent des structures d'ARN. Les relations de deux 2-intervalles $D_1 = (i_1, i'_1)$ et $D_2(i_2 i'_2)$ appartiennent à l'une des trois configuration de la figure 4.13. Un motif de 2-intervalles est défini par un ensemble de 2-intervalles en relation disjointe $\{<, \sqsubset, \between\}$ tel que deux 2-intervalles modélisent l'une des trois relations.

L'auteur s'intéresse au problème PATTERN-MATCHING OVER SET OF 2-INTERVALS consistant à décider de l'existence d'un tel motif dans un ensemble de 2-intervalles donné. Ce problème est proche de celui de la recherche d'ARN structurés. Suivant l'ensemble R des relations autorisées dans le motif, la complexité de ce problème est indiquée dans le tableau 4.2. Le niveau pertinent dans notre cas est le niveau $\{<, \sqsubset, \between\}$ qui est NP-complet. En outre, les relations que nous nous autorisons et qui sont nécessaires pour une modélisation réaliste dépassent largement la famille des trois relations $\{<, \sqsubset, \between\}$ et rendent les résultats de S. Vialette difficilement comparables. Entre autres, nous autorisons des ensemble de 2-intervalles différents à se chevaucher et nous contraignons la distance entre les ensembles de 2-intervalles.

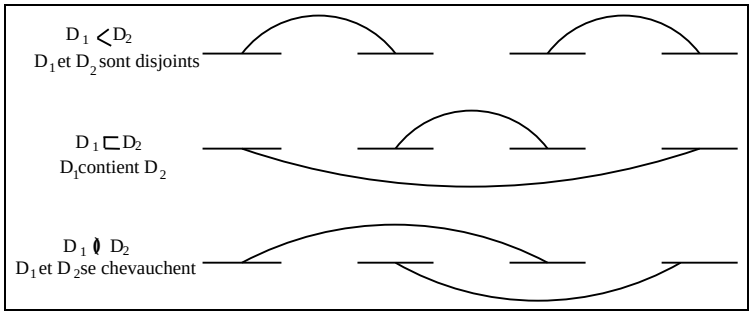


FIG. 4.13 – *Relations modélisées entre deux 2-intervalles (d'après [Via04]).*

Relations R	Recherche du modèle sur un ensemble de 2-intervalles
$\{<, \sqsubset, \between\}$	NP-complet
$\{\sqsubset, \between\}$	NP-complet
$\{<, \sqsubset\}$	$O(mn^3\ log\ n)$
$\{<, \between\}$	$?$
$\{<\}$	$O(n\ log\ n)$
$\{\sqsubset\}$	$O(n^2)$
$\{\between\}$	$O(n^2\ log\ n)$

TAB. 4.2 – *La complexité de la recherche d'un motif dans un famille de* 2*-intervalle avec* $n = |D|$ *et* m *: cardinalité des ensembles de* 2*-intervalles (d'après [Via04]).*

4.3 Conclusion

Notre objectif consiste à proposer une méthodologie pour la recherche de motifs structurés permettant la formulation d'interactions entre molécules (développée dans le chapitre suivant).

Dans cette optique, nous avons envisagé les deux approches décrites au cours de ce chapitre. D'un point de vue résolution, elles ont une complexité théorique équivalente.

Selon l'approche utilisée, la manière d'inférer les caractéristiques des motifs à rechercher est abordée différemment.

Dans la théorie des langages, l'inférence grammaticale est le processus par lequel un programme acquiert la totalité de la structure grammaticale d'un langage sur la base d'exemples positifs et/ou négatifs. Les règles inférées sont en général peu naturelles pour la logique humaine. De plus, il apparaît évident que lorsque l'on ne connaît pas les caractéristiques du motif recherché (ou les caractéristiques d'une famille de motifs), un programme d'apprentissage inférera les règles de grammaire de façon spécifique. Cependant, une quantité importante de données de qualité suffisante est nécessaire pour inférer les relations et la méthode présente, à l'heure actuelle, des limites telles que la prise en compte de pseudo-noeuds qui introduit une complexité supplémentaire (requiert le niveau des grammaires contextuelles).

Dans le cas qui nous intéresse, concernant notamment la recherche d'interactions entre deux molécules indépendantes, le formalisme CSP semble bien adapté. Il permet d'exprimer habilement des relations entre éléments grâce à la puissance expressive et la flexibilité d'emploi le définissant. L'intérêt d'une représentation par contraintes est double. La description des problèmes est purement déclarative, on décrit les solutions du problème et non pas le moyen de calculer les solutions. De plus la formalisation d'une structure par un graphe de contraintes permet d'exprimer simplement des relations, de structure tertiaire par exemple, entre régions sans être limité par la topologie de la molécule. Les caractéristiques des motifs complexes sont exprimées aisément sous la forme de contraintes, quel qu'en soit le niveau de complexité. On peut de plus exprimer les contraintes sous la forme de fonction aussi complexes que l'on veut. En effet, l'ajout de contraintes *ad hoc* ne ruine pas la structure générale du modèle.

Un autre objectif concerne la recherche d'interactions entre motifs structurés indépendants et pouvant être localisés n'importe où dans le texte. La recherche de telles relations, dans le cadre CSP, permet de représenter et traiter facilement le problème. La méthodologie utilisée pour résoudre cette question est abordée dans le chapitre suivant.

Chapitre 5

Formalisme CSP et recherche de motifs structurés

Le chapitre précédent a présenté et comparé deux approches différentes pour répondre au problème posé par la recherche de motifs structurés. Ces deux méthodologies reposent sur deux manières distinctes de modéliser un motif structuré. Nous appelons "motif structuré" l'ensemble formé par une liste ordonnée d'éléments, appelés éléments de séquence. Ces éléments sont considérés comme des objets soumis à des contraintes les reliant entre eux. Le formalisme CSP associé à cette représentation offre l'attrait de privilégier la modélisation, correspondant au point de vue de l'utilisateur, par rapport à la résolution, point de vue du concepteur.

Ce chapitre est consacré à l'adaptation du formalisme CSP dans le cadre particulier de notre problème. Les deux sections suivantes illustrent l'état actuel de nos travaux en décrivant d'une part l'ensemble des éléments que nous souhaitons pouvoir représenter dans ce formalisme et, d'autre part quelques fonctionnalités que nous souhaitons intégrer à notre logiciel de recherche de motifs.

Nous proposons tout d'abord une définition des caractéristiques du motif structuré, pour ensuite proposer une formalisation dans le cadre CSP. Cette formalisation nous permet ainsi de décrire les occurrences que nous souhaitons trouver lors de la recherche de motifs structurés. Muni maintenant d'un cadre CSP, nous nous attacherons dans la section 5.3 à décrire d'un point de vue algorithmique la recherche de solutions du CSP.

5.1 Représentation d'une occurrence

Soit $S = (s_1...s_t)$ un texte de longueur t contenant une chaîne de caractères $(s_1, ..., s_t)$. Un élément de séquence ou mot délimité par les positions (i_1, i_2) contient les caractères $(s_{i1}, ..., s_{i2})$ (par exemple le mot $AGGGCUAGG$ sur la figure 5.1). Un élément de structure constitué de deux éléments de séquence est borné par deux paires de positions $((i_1, i_2), (i_3, i_4))$ (un exemple est donné avec une tige-boucle sur la figure 5.1).

L'objectif de la recherche d'un motif structuré, représenté par un ensemble d'éléments de séquence connectés entre eux, consiste à rechercher l'ensemble des positions des occurrences des éléments.

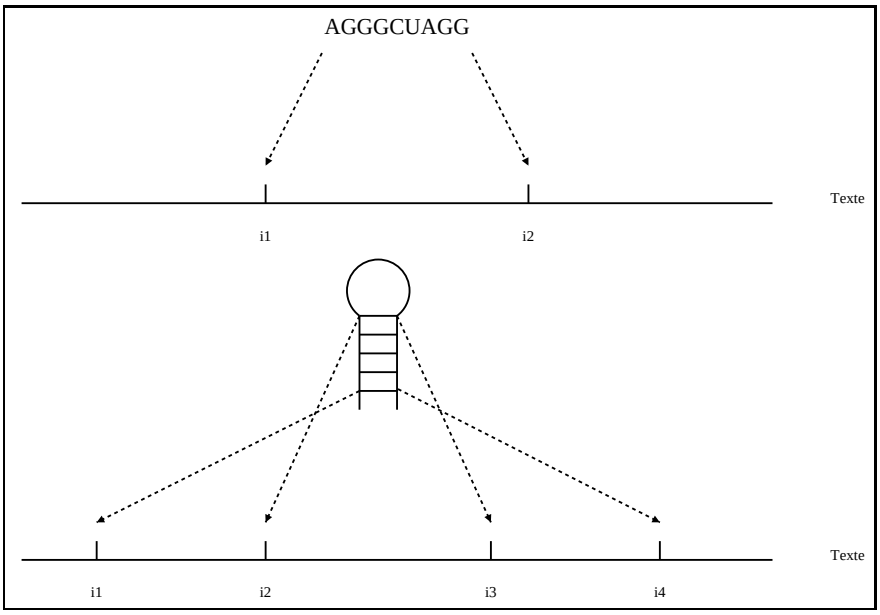


FIG. 5.1 – *Positions correspondant à une occurrence d'un élément de séquence et d'un élément structuré.*

5.1.1 Définitions

Les motifs structurés introduisent la notion de relation entre éléments, laquelle implique une dépendance sur leur contenus. Nous considérons les deux types de relation P_s et R_s entre deux éléments de séquence repérés respectivement sur le texte par les paires d'indices (i_1, i_2) et (i_3, i_4) :

Définition 9 (Relation R_s :)

$$R_s((i_1,i_2),(i_3,i_4)) \textit{ est VRAI } \equiv \begin{cases} i_2 - i_1 = i_4 - i_3 \\ \forall k \textit{ tel que } 0 \leq k < (i_2 - i_1) \\ \quad s_{i_1+k} = s_{i_3+k} \end{cases}$$

Définition 10 (Relation P_s :)

$$P_s((i_1,i_2),(i_3,i_4)) \textit{ est VRAI } \equiv \begin{cases} i_2 - i_1 = i_4 - i_3 \\ \forall k \textit{ tel que } 0 \leq k < (i_2 - i_1) \\ \quad \texttt{compare}(s_{i_4-k}, s_{i_1+k}) \textit{ est VRAI} \end{cases}$$

En utilisant ces deux définitions, nous pouvons décrire un ensemble d'instanciations, représentant les motifs structurés auxquels nous souhaitons nous intéresser. A partir de la relation P_s, les deux éléments de séquences (i_1, i_2) et (i_3, i_4) forment:

un palindrome : si `compare` est la fonction d'identité,

une hélice : si `compare`, pour des appariements de type Watson-Crick et Wobble (par exemple), retourne VRAI pour chaque couple de bases suivant donné en paramètre : (A, U), (U, A), (G, U), (U, G)), (G, C), (C, G)

Les deux éléments de séquences (i_1, i_2) et (i_3, i_4) forment **une répétition** s'il vérifient la relation R_s (9).

Les définitions précédentes peuvent être étendues pour prendre en compte la possibilité d'erreurs lors de la comparaison de caractères. On obtient les relations suivantes:

Définition 11 (Relation $R_{s_{err}}$:)

$$R_{s_{err}}((i_1, i_2), (i_3, i_4), err) \textit{ est VRAI } \equiv \begin{cases} i_2 - i_1 = i_4 - i_3 \\ |r \in \{0...(i_2 - i_1)\} / s_{i_1+r} \neq (s_{i_3+r})| \leq err \end{cases}$$

Définition 12 (Relation $P_{s_{err}}$:)

$$P_{s_{err}}((i_1, i_2), (i_3, i_4), err) \textit{ est VRAI } \equiv \begin{cases} i_2 - i_1 = i_4 - i_3 \\ |r \in \{0...(i_2 - i_1)\} / \texttt{compare}(s_{i_4-r}, s_{i_1+r}) \neq \textit{ VRAI } | \leq err \end{cases}$$

Ces deux définitions permettent de définir des **palindrome, hélice, et répétition** en autorisant des erreurs de type Hamming. Il est également envisageable de définir les erreurs de type Levenshtein.

5.2 Formalisation dans le cadre CSP

Nous définissons les variables d'un CSP à partir des positions des éléments de séquence. Cette nouvelle définition nous permet de formaliser notre problème de recherche de motifs structurés par un problème de satisfaction de contraintes. Le CSP à considérer est un triplet $P = (X, D, C)$ où :

- $X = \{x_1, ...x_n\}$ est un ensemble de n variables $x_{k|k\in 1...n}$. Chaque variable représente une position dans un texte génomique et correspondant à une extrémité d'un élément de séquence composant le motif structuré recherché.
- $D = \{d_1, ..., d_n\}$ représente les domaines de valeurs associés à chacune des variables. Le domaine initial de x_i est $[1, t_i]$ ou t_i est la longueur du texte associé à la variable.
- $C = \{c_1, c_2, ...c_m\}$ est un ensemble de m contraintes sur les variables représentant les relations entre les positions qui délimitent les éléments de séquences.

Une solution du CSP satisfaisant les contraintes posées sur les variables donne une occurrence du motif structuré sous la forme d'un ensemble de positions correspondant aux différents éléments de séquence

La première étape de formalisation du modèle CSP consiste à représenter les différents éléments constituant le motif structuré par un ensemble de variables positions ainsi que par les contraintes les reliant.

La définition des variables et de leurs domaines est évidente, aussi nous nous concentrons dans cette section sur l'ensemble des contraintes que nous pouvons spécifier pour la représentation des occurrences de motifs structurés. Chaque contrainte concerne une ou plusieurs variables

(apparaissant en *italique*) et un ensemble de constantes permettent de la configurer, fixées lors de la modélisation du problème.

5.2.1 Les contraintes unaires

La longueur de l'élément positionné par la variable ainsi que le texte dans lequel est effectuée la recherche sont des paramètres utilisés par les contraintes `Est_De_Nature`, `Contient` et `Est_Composé`.

`Est_De_Nature`(x_i, Texte, l, mot, err, type_{err})

La contrainte est satisfaite quand la position x_i dans Texte correspond à la localisation de mot de longueur l avec au plus err erreurs pour un type d'erreurs donné par type_{err}. La nature des bases autorisées dans un élément fait référence à l'alphabet utilisé.

Variable :

– x_i : représente la position initiale recherchée d'une occurrence du mot dans le texte.

Paramètres :

– Texte : le texte dans lequel le mot est recherché, sur un alphabet à quatre lettres.

– l : la longueur du mot recherché.

– mot : le mot recherché dans le texte. Il peut correspondre à un consensus dégénéré sur un alphabet étendu en utilisant la liste de symboles du code IUPAC (Tab 2.1, page 49).

– err, type_{err} : l'occurrence correspondant au mot recherché peut présenter un certain nombre d'erreurs borné par la valeur err. Le paramètre type_{err} permet de spécifier le type d'erreurs autorisé, spécifiant la distance entre le mot recherché et l'occurrence considérée. Par exemple, une distance de Hamming et/ou de Levenshtein.

`Est_Composé`(x_i, Texte, l, profil, seuil)

La contrainte est satisfaite quand la position x_i dans Texte correspond à la localisation d'un mot de longueur l dont la comparaison des lettres à un profil de fréquence profil donne un score supérieur ou égal à seuil.

Variable :

– x_i : représente la position d'une occurrence d'un mot défini par un profil dans le texte.

Paramètres :

– Texte : le texte dans lequel l'élément de séquence est recherché.

– l : la longueur de l'élément de séquence recherché.

– profil, seuil : un profil de fréquence va donner, pour chaque position, une fréquence pour les caractères A, C, G, T. La comparaison de ce profil à une région du texte donnera un score correspondant à la somme des valeurs de chaque lettre de l'occurrence considérée en fonction de la fréquence donnée par le profil. Le seuil donné en paramètre permet ensuite de rejeter ou non l'occurrence, selon la valeur obtenue lors de la comparaison.

5.2.2 Les contraintes binaires

`Est_Au_Moins_Distant(`$x_{i_1}, x_{i_2}, \mathsf{l}$`)`

La contrainte est satisfaite lorsque la distance entre les deux variables x_{i_1} et x_{i_2} vérifie la relation arithmétique : $x_{i_2} \leq x_{i_1} + \mathsf{l}$.

Variables:

– x_{i_1}, x_{i_2} : les deux positions.

Paramètres:

– l : paramètre longueur minimale contraignant la position d'une variable par rapport à l'autre.

5.2.3 Les contraintes d'arité 4

`Est_Palindrome(`$x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, \mathsf{Texte}, \mathsf{l_{min}}, \mathsf{l_{max}}, \mathsf{b_{min}}, \mathsf{b_{max}}, \mathsf{regle}, \mathsf{err}, \mathsf{type_{err}}$`)`

La contrainte est satisfaite lorsque les mots délimités par les couples (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}) sur Texte vérifient une relation de type P_s (Fig 10) avec la règle regle et au plus err erreurs. Ces deux mots ont une certaine longueur appartenant à l'intervalle $[\mathsf{l_{min}}, \mathsf{l_{max}}]$ et sont distant d'une valeur appartenant au second intervalle $[\mathsf{b_{min}}, \mathsf{b_{max}}]$.

Variables:

– $x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}$: les quatre positions bornant les deux régions (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}) impliquées dans un palindrome.

Paramètres:

– Texte : le texte dans lequel le palindrome est recherché.

– $\mathsf{l_{min}}, \mathsf{l_{max}}, \mathsf{b_{min}}, \mathsf{b_{max}}$: ces quatre paramètres contraignent les distances entre chaque paire de variables. Si l'on se réfère à la figure 5.2 où un palindrome entre deux régions non contigües est représenté sous la forme "dépliée", on désigne par $\mathsf{l_{min}}, \mathsf{l_{max}}$ les longueurs minimum et maximum des couples (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}), correspondant à la taille de "bras" du palindrome. $\mathsf{b_{min}}$ et $\mathsf{b_{max}}$ restreignent l'intervalle de longueur de la région (on parlera de "boucle") devant séparer les deux "bras", soit la distance entre $(x_{i_2,}, x_{i_3})$.

– regle : le paramètre de règle donne la nature de la fonction $\mathsf{compare}$ utilisée par la relation P_s. Si la règle utilisée correspond à l'égalité, le motif structuré correspond à un palindrome au sens strict. En la définissant par une relation de complémentarité on peut alors représenter une hélice.

– $\mathsf{err}, \mathsf{type_{err}}$: la prise en compte d'erreurs est permise par la relation $P_{s_{err}}$. Le nombre d'erreurs autorisées dans le motif structuré est borné par la valeur err. Le paramètre $\mathsf{type_{err}}$ permet de spécifier le type d'erreurs autorisé par la relation, lequel définit la distance entre le modèle décrit et l'occurrence considérée. Il est ainsi possible de n'autoriser que des distances de Hamming ou des distances de Hamming et Levenshtein. Dans le premier cas de figure, la distance entre les couples de variables (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}) sera identique et comprise dans les intervalles spécifiés. Dans le second cas, où des "insertions" ou "délétions" de caractères sont autorisées dans les deux "bras", les distances entre (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}), toujours comprises dans les intervalles spécifiés, pourront être différentes.

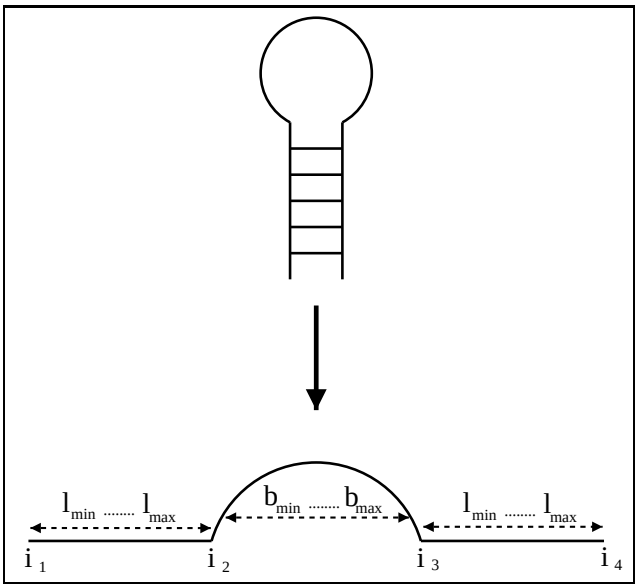


FIG. 5.2 – *Un motif structuré représenté sous forme dépliée.*

`Est_Répétition` ($x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, \mathsf{Texte}, \mathsf{l_{min}}, \mathsf{l_{max}}, \mathsf{b_{min}}, \mathsf{b_{max}}, \mathsf{err}, \mathsf{type_{err}}$)

La contrainte est satisfaite lorsque les mots délimités par les couples (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}) sur Texte vérifient une relation de type R_s avec au plus err erreurs. Ces deux mots ont une longueur appartenant à l'intervalle $[\mathsf{l_{min}}, \mathsf{l_{max}}]$ et sont distant d'une valeur appartenant au second intervalle $[\mathsf{b_{min}}, \mathsf{b_{max}}]$.

Les paramètres utilisés par la contrainte sont identiques à ceux décrits précédemment pour la contrainte `Est_Palindrome`. La seule différence concerne le paramètre regle qui dans le cas présent est inutile.

`Est_Duplex` ($x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, \mathsf{Texte_1}, \mathsf{Texte_2}, \mathsf{l_{min}}, \mathsf{l_{max}}, \mathsf{err}, \mathsf{type_{err}}$)

La contrainte est satisfaite lorsque les mots délimités par les couples (x_{i_1}, x_{i_2}) sur $\mathsf{Texte_1}$ et (x_{i_3}, x_{i_4}) sur $\mathsf{Texte_2}$ vérifient une relation de type P_s (Fig 9) avec la règle regle et au plus err erreurs. Ces deux mots, sur deux textes différents, ont une certaine longueur appartenant à l'intervalle $[\mathsf{l_{min}}, \mathsf{l_{max}}]$.

Les paramètres utilisés par la contrainte sont identiques à ceux décrits précédemment pour la contrainte `Est_Palindrome`, à l'exception de celui concernant le texte et de ceux donnant l'intervalle de longueur devant séparer les deux couples (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}). La distance (x_{i_2}, x_{i_3}) n'a plus de sens ici.

`Ont_Une_Taille`($x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}$)

Elle définit la distance entre les couples (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}) par l'expression arithmétique: $|x_{i_2} - x_{i_1}| = |x_{i_4} - x_{i_3}|$

La section 5.4 présente le processus de conversion de ces contraintes biologiques en contraintes fonctionnelles pour le CSP.

5.2.4 Quelques exemples

A partir de ces contraintes, nous pouvons représenter un certain nombre de motifs biologiques décrits dans le premier chapitre en délimitant certains éléments de séquence par des variables et en utilisant les contraintes précédemment définies.

- Dans l'exemple (1) de la figure 5.3 est représentée une tige-boucle suivie d'un mot. Une première contrainte `Est_Hélice` implique les éléments positionnés par les couples de variables $((x_1, x_2), (x_3, x_4))$ utilisant la relation P_s, et une seconde contrainte `Est_De_Nature` sur x_4 détermine le mot qui devra commencer à cette position.
- Dans l'exemple (2) de la figure 5.3 est représenté un pseudo-noeud où une première contrainte `Est_Hélice` est posée sur les couples de variables $((x_1, x_2), (x_5, x_6))$ (avec la contrainte d'ordre implicite $(x_4 < x_5)$ et $(x_3 > x_2)$) et une seconde contrainte de même type sur les couples de variables $((x_3, x_4), (x_7, x_8))$.
- Dans l'exemple (3) de la figure 5.3 est représentée une triple hélice où une première contrainte `Est_Hélice` est posée sur les couples de variables $((x_1, x_2), (x_3, x_4))$ et une seconde contrainte de même type sur les couples de variables $((x_3, x_4), (x_5, x_6))$.
- Dans l'exemple (4) de la figure 5.3 est représentée une quadruple hélice où trois contraintes `Est_Répétition` sont posées sur les couples de variables $((x_1, x_2), (x_3, x_4))$, $((x_3, x_4), (x_5, x_6))$ et $((x_5, x_6), (x_7, x_8))$.

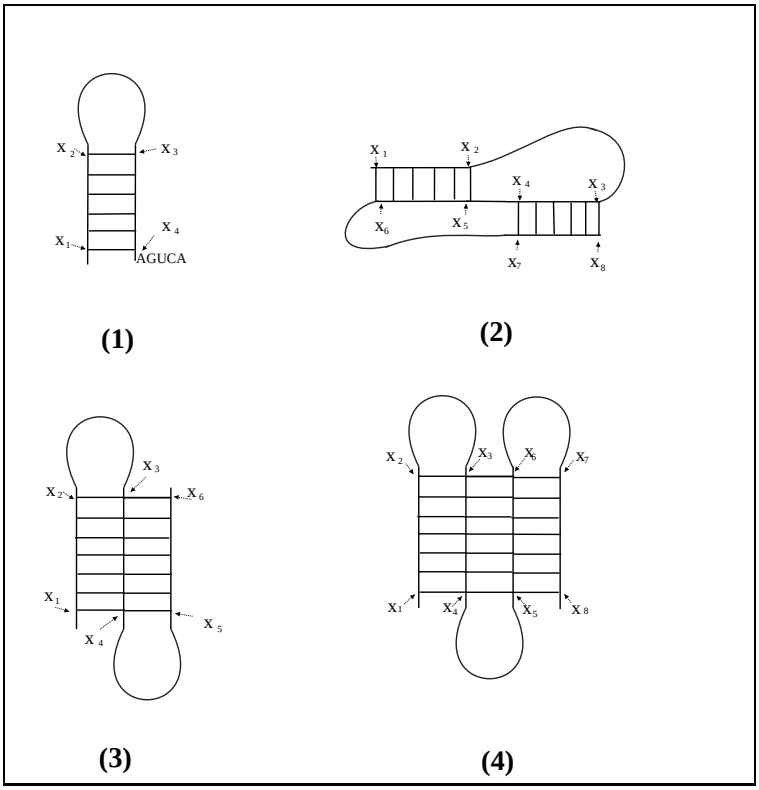


FIG. 5.3 – *Représentation de motifs structurés : (1) une tige-boucle suivi d'un mot, (2) un pseudo-noeud, (3) une triple hélice et (4) une quadruple hélice.*

5.3 La recherche de solutions

A partir de la formalisation du problème de recherche de motifs structurés dans le cadre CSP, nous sommes maintenant en mesure d'utiliser les algorithmes associés aux CSP et présentés dans le chapitre précédent. Cette recherche de solutions consiste à réduire le CSP initial en un CSP équivalent mais plus simple en réduisant les domaines par vérification de la cohérence locale (arc-cohérence, arc-cohérence aux bornes) et en l'associant à un parcours récursif de l'arbre de recherche (`retour-arrière`).

Nous présentons dans une première partie la procédure d'exploration de l'arbre de recherche et nous détaillons par la suite les différentes procédures utilisées pour réduire les domaines de valeurs des variables explorées, lesquelles sont liées à la nature spécifique de chaque contrainte.

5.3.1 Choix de la méthode de filtrage

Le domaine de valeurs associé à chaque variable comprend l'ensemble des entiers compris dans l'intervalle $[1, t]$, t étant la longueur du texte où la recherche est effectuée (certaines variables peuvent prendre leurs valeurs dans un intervalle différent $[1, u]$ lorsqu'un second texte de longueur u est donné).

Les techniques de cohérence locale ont été appliquées avec succès aux CSP et peuvent être très utiles pour les systèmes de contraintes numériques. Cependant l'application de techniques de cohérence locale simples telles que l'arc-cohérence peut également devenir irréalisable sur des domaines de valeurs très grands [Lho93]. Dans notre problème, la taille du texte correspondant à un génome, peut être de l'ordre de plusieurs millions de bases. L'arc-cohérence pour un tel cas de figure n'était guère envisageable (trop coûteuse en espace et en temps) et nous avons utilisé de ce fait les techniques de propagation des bornes d'intervalles. Elles consistent à propager les bornes des domaines et sont souvent utilisées dans l'implémentation de CSP avec des domaines d'entiers. Intuitivement, la $2B$-Cohérence est une approximation de la cohérence d'arc.

Définition 13 ($2B$-Cohérence) *:*
Un CSP est $2B$-cohérent [BO97, Lho93] si et seulement si toutes ses contraintes sont $2B$-cohérentes. Une contrainte c est $2B$-cohérente, si et seulement si, pour toute variable $x_i \in$ l'ensemble des variables de c de domaine $D_x = [a_i, b_i]$, les deux instanciations partielles (x_i, a_i) et (x_i, b_i) peuvent être prolongées en instanciations globales satisfaisant le CSP.

Un CSP est $2B$-cohérent si toutes ses contraintes sont $2B$-cohérentes.

La $2B$-cohérence est une cohérence plus faible que la cohérence d'arc :

- la cohérence d'arc impose une condition sur tous les éléments des domaines;
- la $2B$-cohérence impose cette même condition aux seules bornes de l'intervalle qui contient les domaines.

Exemple

$P = \{\{x, y\}, \{D_x, D_y\}, \{x = y^2\}\}$ avec $D_x = [1, 4]$, $Dy = [-2, +2]$.

P est $2B$-cohérent: les valeurs $x = 1$ et $x = 4$ admettent des valeurs de y compatibles ($y = 1$ et $y = 2$ ou $y = -2$, respectivement), les valeurs $y = -2$ et $y = 2$ admettent également

des valeurs compatibles ($x = 4$ dans les deux cas). Par contre les valeurs $x = 2$ et $x = 3$ n'admettent pas de valeurs entières de y compatibles, de même, la valeur $y = 0$ n'admet pas de valeur du domaine D_x compatible, P n'est donc pas arc-cohérent.

5.3.2 Mécanisme général de résolution du CSP

Nous allons nous intéresser plus particulièrement dans cette section au mécanisme général de résolution de contraintes. Il s'appuie sur une méthode de résolution hybride basée sur une exploration arborescente de type "retour-arrière" tout en établissant à chaque noeud de l'arbre une certaine cohérence locale (ici la $2B$-cohérence pour des raisons d'efficacité). A chaque noeud, si l'établissement de la $2B$-cohérence détecte l'incohérence du problème, un retour arrière a lieu. Sinon, la recherche de solution progresse avec un problème simplifié.

D'un point de vue opérationnel, le système d'établissement de la cohérence locale utilisé est un système classique en programmation par contraintes s'appuyant sur la notion d'événements et de "démons". Lors de l'établissement de la cohérence d'arc classique, le seul événement pouvant demander l'application de la fonction `Revise`, destinée à rétablir la cohérence d'arc, est l'effacement de valeurs. Dans notre cas, le domaine de chaque variable x_i est représenté par un intervalle $[v_{bi}, v_{bs}]$ et ce sont les modifications de la borne inférieure ou supérieure de l'intervalle qui constitue un événement nécessitant un rétablissement de la $2B$-cohérence.

Pour détecter ces événements, toute tentative de la modification de la borne inférieure (respectivement supérieure) d'une variable s'effectue par une procédure dédiée `Est`$\geq$, acceptant en paramètre une variable et un nouveau minorant de cette variable (déduit par filtrage ou instanciation d'une variable). La procédure est décrite dans la figure 5.4 (respectivement `Est`$\leq$, `Fig 5.5`). Si le nouveau minorant n'améliore pas la borne inférieure, rien ne se passe. Sinon, toutes les contraintes qui impliquent cette variable sont notifiées de l'événement par l'appel d'une procédure de filtrage, spécifique à chaque contrainte appelée procédure `ReviseBorneInférieure` et `ReviseBorneSuperieure`, suivant le type de l'événement. On obtient ainsi une souplesse extrême dans la propagation des contraintes: chaque type de contrainte peut utiliser une méthode de filtrage *ad-hoc*. La propriété essentielle que doit satisfaire une telle méthode est de simplifier le problème (supprimer des valeurs compatibles) et de détecter la violation de contraintes si les variables sont instanciées.

En pratique, ces fonctions de filtrage spécifiques à chaque contrainte sont en général chargées de restaurer la $2B$-cohérence en éliminant des extrémités des domaines des autres variables celles qui n'ont plus de support sur la contrainte suite à l'événement de modification. Ces éventuelles modifications des bornes des autres variables peuvent à leur tour déclencher d'autres événements, entraînant le réveil de nouvelles méthodes de filtrage et ainsi, jusqu'à stabilisation (atteinte d'un point fixe, garantie dans notre cas du fait de la finitude des domaines).

Algorithme : Est$\geq(x_i, val)$
si $val > v_{bi}$ **alors**
  si $val > v_{bs}$ **alors**
    Exception;
  sinon
    $v_{bi} \leftarrow val$
    pour chaque c_k *de* $LISTE_i$ **faire**
      ReviseBorneInférieure(c_k, x_i);

FIG. 5.4 – *Procédure* Est$\geq$.

Algorithme : Est$\leq(x_i, val)$
si $val < v_{bs}$ **alors**
  si $val < v_{bi}$ **alors**
    Exception
  sinon
    $v_{bs} \leftarrow val$
    pour chaque c_k *de* $LISTE_i$ **faire**
      ReviseBorneSupérieure(c_k, x_i)

FIG. 5.5 – *Procédure* Est$\leq$.

Dans notre implémentation, la gestion des événements est en "profondeur d'abord": si un événement e_1 génère, via l'application d'un filtrage un événement e_2, alors e_2 est immédiatement exploité pour déclencher de nouveaux filtrages sans que e_1 soit totalement exploité. Cette méthode s'implémente simplement en entreposant les événements dans une pile. En pratique c'est la pile d'appel C qui est exploitée. Plusieurs événements identiques peuvent se trouver simultanément dans la pile.

Si un domaine est vide après une série d'événements et de filtrages induits par une instanciation de variable par exemple (Fig 5.6), une exception est levée par les méthodes de détection d'événement Est$\geq$ et Est$\leq$. Cette exception est ensuite capturée et traitée par l'algorithme d'exploration arborescente.

Algorithme : `retour-arrière` (i)

/*i : indice de la variable courante*/
si $i = n$ */*nombre de variables*/* **alors**
  /* On a donné une valeur à toutes les variables avec succès */
  Une solution a été trouvée
sinon
  si $v_{bi} = v_{bs}$ */* la variable n'a qu'une valeur, on passe à la suite*/* **alors**
    `retour-arrière` $(i+1)$
  sinon
    tant que $v_{bi} < v_{bs}$ **faire**
      $val \leftarrow v_{bi}$
      Sauvegarde des bornes de toutes les variables $x_i ... x_n$
      */*On essaie d'affecter val à x_i et en cas d'échec on récupère une exception*/*
      Essai
        `Est`$\geq(x_i, v_{bi})$
        `Est`$\leq(x_i, v_{bi})$
        `retour-arrière` $(i+1)$
      Capture exception
        /*on retourne à $i-1$*/
      Restoration des bornes de toutes les variables $x_i ... x_n$
      `Est`$\geq(x_i, v_{bi}$`+1 )` /* on passe à la valeur suivante*/
  si $v_{bi} = v_{bs}$ */* si le domaine est un singleton, on passe directement à la variable suivante*/* **alors**
    `retour-arrière` $(i+1)$

Procédure faisant le premier appel de `retour-arrière`**.**

Essai
  `retour-arrière` (1)
Capture exception
  si *i = 0* **alors**
    Arrêt de la recherche : Incohérence

FIG. 5.6 – *Procédure de* `retour-arrière`*.* i *indice de la variable courante,* x_i *: variable courante avec son domaine d(*x_i*) =* $[v_{bi}, v_{bs}]$*, respectivement la borne inférieure et supérieure.*

5.4 La propagation des contraintes

Le formalisme CSP permet de considérer séparément chaque contrainte lors de l'implémentation. Dans le cadre de notre problème, les contraintes sont utilisées pour générer à la volée les valeurs des variables satisfaisant localement les motifs les impliquant. Les différentes règles de propagation sont l'application directe de leur relation algébrique ou la mise en oeuvre d'approches issues de l'algorithmique de texte. L'objectif de la propagation de ces contraintes est, pour un domaine courant d'une variable, de resserrer les bornes du domaine afin d'atteindre la cohérence de bornes. A cette fin, nous définissons des règles de propagation appropriées pour chaque type de contrainte. Les techniques de cohérence locale utilisent ensuite ces règles de propagation pour mettre à jour le domaine de la variable de réveil.

Cette section a pour vocation de décrire en détail certaines contraintes présentées précédemment, auxquelles nous nous sommes intéressés:

Contraintes algébriques : la relation entre les variables est algébrique. Selon le nombre de variables impliquées, on distingue :

- Contrainte `Est_Au_Moins_Distant` porte sur deux variables. L'entier délimitant la distance est statique et est donné lors de la modélisation du CSP.
- Contrainte `Ont_Une_Taille` porte sur quatre variables. Les entiers délimitant les intervalles de longueur sont dynamiques et sont donnés par le calcul de la longueur des bornes de deux variables.

Contraintes spécifiques : elles spécifient les mots dans le texte positionnés par une ou plusieurs variables par l'application d'une règle de propagation. Nous distinguons trois niveaux de complexité:

- Contrainte `Contient` porte sur une variable. La règle de propagation associée fait appel à un algorithme de recherche de mot. Le mot recherché est statique et donné lors de la modélisation du problème.
- Contraintes `Est_Palindrome` et `Est_Répétition` portent sur quatre variables. La règle de propagation associée fait appel à un algorithme de recherche de motifs structurés. Le mot recherché est dynamique.
- Contrainte `Est_Duplex` porte sur quatre variables. La règle de propagation associée fait appel à un algorithme de recherche de mot lequel utilise une structure de données de type arbre des suffixes. Le mot recherché est dynamique.

Les complexités des règles de propagation de chacune de ces contraintes sont différentes. En effet, effectuer une recherche de mots dans un texte n'est pas du même ordre qu'effectuer une comparaison d'entiers. Du fait de la généricité de l'application des contraintes et du maintien de l'arc cohérence, les contraintes de type arithmétiques ou de type "recherche de mot" sont traitées de la même manière. Cependant, les domaines des variables sont rapidement rétrécis par les contraintes de distance et permettent aux contraintes de "recherche de mot" de travailler plus rapidement.

De plus, la mémorisation des dernières valeurs localement cohérentes économise dans certains cas l'appel des algorithmes coûteux de recherche de mot. La contrainte commence par vérifier si les dernières valeurs trouvées et sauvegardées par la fonction de recherche de mot

lors du dernier appel de la contrainte n'est pas dans le domaine actuel de la variable. Si tel est le cas, la valeur sauvegardée val_{save} est obligatoirement (liée au raisonnement par bornes) la première position satisfaisant la contrainte dans le domaine courant de la variable. De ce fait, la contrainte fait l'économie d'un appel à la fonction de recherche de mot.

Nous présentons en détail dans la section suivante quelques unes de ces contraintes.

5.4.1 Contrainte `Est_Au_Moins_Distant`

Le couple de variables impliquées dans la contrainte `Est_Au_Moins_Distant` est (x_1 ,x_2) ayant respectivement comme bornes pour leurs domaines de valeurs : $[v_{1_{bi}}, v_{1_{bs}}]$ et $[v_{2_{bi}}, v_{2_{bs}}]$.

5.4.1.1 Fonction de propagation

L'expression algébrique de la contrainte est : $x_{i_2} \geq x_{i_1} + \delta$. Son application essaie de réduire les domaines des deux variables ($x_{i_1} = [v_{1_{bi}}, v_{1_{bs}}]$ et $x_{i_2} = [v_{2_{bi}}, v_{2_{bs}}]$) de la manière suivante:

- augmenter la valeur de la borne inférieure $v_{2_{bi}}$ de x_{i_2} pour que : $v_{2_{bi}} \geq v_{1_{bi}} + \delta$
- diminuer la valeur de la borne supérieure de $v_{1_{bs}}$ de x_{i_1} pour que : $v_{1_{bs}} \leq v_{2_{bs}} - \delta$

5.4.1.2 Mécanisme de propagation

Si la contrainte est réveillée par une modification de la borne inférieure $v_{1_{bi}}$ du domaine de valeurs de x_1 alors la propagation est effectuée sur la borne inférieure $v_{2_{bi}}$ de x_2 . Si le réveil est lié à une modification de la borne supérieure $v_{2_{bs}}$ du domaine de valeurs de x_2, la propagation se fait sur la borne supérieure $v_{1_{bs}}$ de x_1.

5.4.1.3 Extension de la contrainte

Contraindre deux variables par un intervalle $[\delta_{min}, \delta_{max}]$ se réalise par la spécification des deux contraintes suivantes:
`Est_Au_Moins_Distant`($x_1, x_2, \delta_{\mathsf{min}}$)
`Est_Au_Moins_Distant`($x_2, x_1, -\delta_{\mathsf{max}}$).

5.4.2 Contrainte `Est_De_Nature`

`Est_De_Nature`(x, Texte, I, mot, err, $\mathsf{type}_{\mathsf{err}}$) permet de spécifier la nature de certaines bases à une position donnée. Le mot recherché est connu *a priori* et est spécifié lors de la modélisation du problème.

5.4.2.1 Fonction de propagation

La fonction de propagation appelée par la contrainte est :
`fonction_recherche_de_mot`(x, Texte, I, mot, err, $\mathsf{type}_{\mathsf{err}}$).

A partir de la région du texte, donné en paramètre Texte, délimité par les bornes $[v_{bi}, v_{bs}]$ du domaine de valeurs de x, la fonction cherche la position minimale d'une occurrence satisfaisante. En cas de succès, elle renvoie sa position initiale (val) et en cas d'échec (la fonction a atteint la position v_{bs} sans trouver le mot), elle retourne la valeur -1.

Dans le chapitre 2 nous décrivons deux familles d'algorithmes de recherche de motifs dont les avantages de chacune sont liés à la connaissance préalable ou non du mot recherché. Dans le cas présent, effectuer un pré-traitement lors de la construction du réseau du CSP sur le mot recherché semble la meilleure manière de procéder. Nous avons ainsi implémenté l'algorithme de Manber (décrit dans la section 2.2.3, 54) pour effectuer la recherche de mot.

Nous avons également testé l'utilisation de l'algorithme Boyer-Moore, mais l'utilisation de l'algorithme de Manber nous permet de traiter à la fois la prise en compte de distance entre l'occurrence et le mot recherché par la distance de Hamming et de Levenshtein, et d'utiliser le code IUPAC. L'algorithme Boyer-Moore n'a dons pas été utilisé par la suite.

5.4.2.2 Mécanisme de propagation

Pour une première implémentation, la propagation de la contrainte n'est effectuée qu'à partir du réveil de la borne inférieure du domaine de la variable. Il est possible de rendre la propagation de cette contrainte symétrique en effectuant la recherche du mot inversé à partir de la borne supérieure et d'améliorer ainsi l'efficacité de la propagation.

Dans le cas où la valeur mémorisée par la contrainte n'est pas dans le domaine courant de la variable, la fonction de recherche de mot est appelée. Si elle renvoie la valeur -1, la valeur $v_{bi} + 1$ (en dehors du domaine de la variable) lui est affectée. Ainsi, lors du lancement de la mécanique de propagation à partir de la procédure `Est`$\geq$, la gestion de l'échec de la recherche correspond à une exception qui permet alors d'arrêter l'exploration inutile de l'arborescence correspondante. En cas de succès la borne inférieure du domaine de la variable est réduit.

Algorithme : `ReviseBorneInférieure`

```
si v_bi > val_save alors
    val ← fonction_recherche_de_mot(x, Texte, l, mot, err, type_err)
    si val = −1 /* la recherche a échouée*/ alors
        val ← v_bs + 1
    sinon
        val_save ← val /*Sauvegarde de la valeur*/
    Est≥(x, val)
```

FIG. 5.7 – `Procédure ReviseBorneInférieure`.

5.4.2.3 Exemple

L'exemple suivant illustre la formalisation d'un CSP à partir d'un motif composé de deux mots (Fig 5.8) éloignés par une certaine distance. La formalisation du CSP (Fig 5.9) nécessite deux variables délimitant chaque début de mot, chacune étant impliquée dans une contrainte unaire `Est_De_Nature`. La position de chacune des deux variables est également liée par l'utilisation de deux contraintes `Est_Au_Moins_Distant`. Ce CSP est représenté graphiquement par la figure 5.10 où les variables et leur domaines sont représentés par une ellipse, la contrainte de distance les reliant par une ligne horizontale, et chaque contrainte sur le mot par un arc de cercle ayant pour origine et fin la variable concernée.

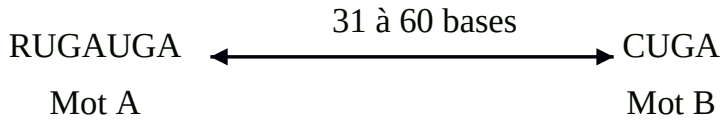


FIG. 5.8 – *Description du motif recherché.*

$$\begin{cases} V = \{x_1, x_2\} \\ D = \begin{cases} D_1 = [1, t] \\ D_2 = [1, t] \end{cases} \\ Contraintes = \begin{cases} \texttt{Est\_De\_Nature}(x_1, \mathsf{Texte}, 7, \text{''}\,\mathsf{RUGAUGA}\text{''}, 0, 0) \\ \texttt{Est\_De\_Nature}(x_2, \mathsf{Texte}, 4, \text{''}\,\mathsf{CUGA}\text{''}, 0, 0) \\ \texttt{Est\_Au\_Moins\_Distant}(x_1, x_2, 7 + 31) \\ \texttt{Est\_Au\_Moins\_Distant}(x_2, x_1, -7 - 60) \end{cases} \end{cases}$$

FIG. 5.9 – *Formalisation du problème dans le cadre CSP.*

FIG. 5.10 – *Représentation en réseau du CSP correspondant au motif recherché.*

5.4.3 Contrainte `Est_Palindrome`

`Est_Palindrome`$(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, \mathsf{Texte}, \mathsf{l_{min}}, \mathsf{l_{max}}, \mathsf{b_{min}}, \mathsf{b_{max}}, \mathsf{regle}, \mathsf{err}, \mathsf{type_{err}})$ définit une relation de type palindrome.

5.4.3.1 Fonction de propagation

La recherche d'un palindrome d'une longueur pouvant varier dans $[\mathsf{l_{min}}, \mathsf{l_{max}}]$ dans un texte de longueur t consiste à identifier deux régions dont chaque caractère vérifie la relation P_s. Cette

$$\begin{cases} j_1 = MAX \begin{cases} v_{1_{bi}} \\ v_{2_{bi}} - \mathsf{l_{max}} \\ v_{3_{bi}} - \mathsf{l_{max}} - \mathsf{b_{max}} \\ v_{4_{bi}} - (2 \times \mathsf{l_{max}}) - \mathsf{b_{max}} \end{cases} \\ j'_1 = MIN \begin{cases} v_{1_{bs}} \\ v_{2_{bs}} - \mathsf{l_{min}} \\ v_{3_{bs}} - \mathsf{l_{min}} - \mathsf{b_{min}} \\ v_{4_{bs}} - (2 \times \mathsf{l_{min}}) - \mathsf{b_{min}} \end{cases} \end{cases}$$

$$\begin{cases} j_3 = MAX \begin{cases} j_1 + (2 \times \mathsf{l_{min}}) + \mathsf{b_{min}} \\ v_{2_{bi}} + \mathsf{l_{min}} + \mathsf{b_{min}} \\ v_{3_{bi}} + \mathsf{l_{min}} \\ v_{4_{bi}} \end{cases} \\ j'_3 = MIN \begin{cases} j_1 + (2 \times \mathsf{l_{max}}) + \mathsf{b_{max}} \\ v_{2_{bs}} + \mathsf{l_{max}} + \mathsf{b_{max}} \\ v_{3_{bs}} + \mathsf{l_{max}} \\ v_{4_{bs}} \end{cases} \end{cases}$$

FIG. 5.11 – *Procédure de calcul des intervalles de recherche de chaque bras du palindrome, utilisés par l'algorithme de la figure 5.12.* $v_{i_{bi}}$ *correspond à la borne inférieure du domaine de* x_i *et* $v_{i_{bs}}$ *correspond à la borne supérieure.*

recherche peut être initiée à partir de chacune des quatre variables. Elle a pour objectif de chercher le premier palindrome minimisant la valeur prise par la variable concernée. Cette approche nous permet ainsi grâce à la mécanique de la $2B$-cohérence d'effectuer une recherche exhaustive. Nous décrivons la fonction `Est_palindrome_x1` (Fig 5.12) dédiée à la recherche de palindromes effectuées à partir de la borne inférieure de x_1. La première étape consiste à calculer l'intervalle précis (j_1, j'_1) correspondant à la région où l'amorce du palindrome peut être effectuée. Le second intervalle (j_3, j'_3) correspond à la région où la recherche du second "bras" pour une valeur donnée de j_1 doit être initiée. Pour une meilleur clarté, nous présentons séparément les procédures des calculs de (j_1, j'_1) et (j_3, j'_3) (Fig 5.11) utilisées par l'algorithme de `Est_palindrome_x1`.

La fonction de comparaison de caractères `compare` prend en paramètre le texte où le palindrome est recherché, les deux positions d'intérêt et la règle choisie pour effectuer la comparaison. En cas de succès la fonction retourne la valeur nulle, et en cas d'échec la valeur 1. Le nombre de mésappariements est alors testé et s'il est inférieur à la valeur err correspondant au nombre total de mésappariements autorisés par la contrainte, l'extension de l'hélice peut continuer. La fonction `Est_palindrome_x1` retourne en cas de succès la première valeur de x_1 où un palindrome satisfait les contraintes de longueurs et d'appariements. En cas d'échec, lorsque toutes les valeurs possibles pour i ont été testées, elle renvoie -1.

Une autre fonction, inspirée de la programmation dynamique, permet de réaliser la recherche de palindrome en gérant les insertions et deletions.

```
Algorithme : Est _palindrome_x1
(x_{i1}, x_{i2}, x_{i3}, x_{i4}, Texte, l_min, l_max, b_min, b_max, regle, err, type_err)
H : longueur de l'hélice;
M : nombre de mésappariements;
Calcul de j_1 et j'_1;
pour chaque j_1 ≤ i ≤ j'_1 faire
    Calcul de j_3 et j'_3;
    tant que k ≤ j'_3 faire
        M=0;
        H=0;
        i' ← i;
        k' ← k;
        tant que ((M ← M + compare(Texte, i', k', regle)) ≤ err et H < l_min)
        faire
            H ← H + 1;
            i' ← i' + 1;
            k' ← k' − 1;
        si H ≥ l_min alors
            retourner [i]
        k ← k + 1 ;
retourner [-1]
```

FIG. 5.12 – `Est _palindrome_x1`. *Les procédures de calculs de j_1, j'_1, j_3, j'_3 sont données par la figure 5.11.*

5.4.3.2 Mécanisme de propagation

La fonction `ReviseBorne` (Fig 5.13) effectue la recherche de palindromes à partir de chacune des quatre variables.

Pour chaque recherche de palindrome initiée à partir d'une variable x_i , afin d'éviter une recherche inutile, la contrainte commence par vérifier:

- si la dernière valeur trouvée par la fonction de recherche de palindrome est dans le domaine actuel de la variable
- et si les bornes mémorisées des domaines des trois autres variables appartiennent à leur domaine courant

Si tel est le cas, la valeur sauvegardée $val_{i_{save}}$ est obligatoirement (raisonnement par bornes) la première position satisfaisant la contrainte pour le domaine courant de la variable x_i. De ce fait, la contrainte fait l'économie d'un appel à la fonction en restant silencieuse et ne déclenche pas inutilement la mécanique de propagation.

Si le test est négatif, alors la recherche de palindrome est réalisée et la valeur correspondant

au premier palindrome pour x_i est propagée.

Algorithme : ReviseBorne

```
pour chaque 1 ≤ i ≤ 4 faire
    si v_{i_bi} > val_{i_save} alors
        val_i ← Est_helice(x_i, Texte, l_min, l_max, b_min, b_max, regle, err, type_err) si
        val_i = −1 /* la recherche a échouée*/ alors
            val ← v_{i_bs} + 1 ;
        sinon
            val_{i_save} ← val_i
        Est≥(x_i, val)
```

FIG. 5.13 – Procédure ReviseBorne.

La complexité d'une approche naïve, lorsque les deux régions ne sont pas contraintes par une certaine distance est quadratique. La première étape consiste à parcourir les t caractères du texte avec un pas de $+1$ et pour chacun d'eux à procéder à la comparaison de chaque caractère du texte avec un pas de $+1$. Cette première phase consiste à rechercher une amorce et présente une complexité en $O(\mathsf{t} \times \mathsf{t})$. Lorsqu'une des comparaisons retourne vrai, la seconde étape consiste à tenter l'extension du palindrome et donne une complexité globale de $O(t \times t \times \mathsf{l_{max}})$. Grâce à la prise en compte des paramètres de distance $[\mathsf{b_{min}}, \mathsf{b_{max}}]$ autorisé entre les deux régions corrélées, la recherche de l'amorce pour chaque caractère du texte n'est plus effectuée sur la totalité du texte mais sur une région précise. La complexité de l'algorithme devient de ce fait dans le pire des cas : $O(t \times (\mathsf{l_{max}} + \mathsf{b_{max}} - \mathsf{l_{min}} - \mathsf{b_{min}}) \times \mathsf{l_{max}})$ donc linéaire dans la taille du texte.

En pratique, c'est le domaine de x qui est utilisé au lieu de $[1, t]$ et il est très rapidement réduit après instanciation d'une variable et après la propagation des contraintes de distance.

5.4.3.3 Exemple

L'exemple suivant illustre la formalisation d'un CSP à partir d'un motif structuré de type tige-boucle (Fig (1) 5.14). La formalisation du CSP (Fig 5.15) nécessite quatre variables délimitant chacune des deux régions symétriques. La position de chaque couple de variables, délimitant chaque "bras", est liée par l'utilisation de deux contraintes `Est_Distant` (une variante de `Est_Au_Moins_Distant`) et le couple bornant la boucle par deux contraintes `Est_Au_Moins_Distant`. Les quatre variables sont également reliées par une contrainte `Est_Palindrome`.

Ce CSP est représenté graphiquement par la figure (2) 5.14 où les variables et leur domaines sont représentés par un cercle, les contrainte de distance les reliant par une ligne horizontale ou verticale, et la contrainte d'arité 4 par une croix avec un petit rectangle au centre.

La Figure 5.16 est une sortie de programme et montre l'application de la cohérence locale aux bornes des domaines des variables. Les trois solutions chevauchantes sont trouvées.

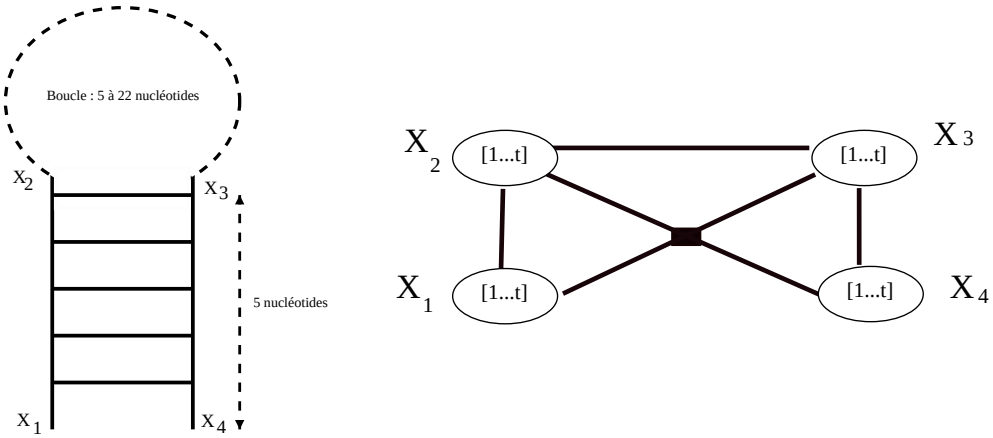
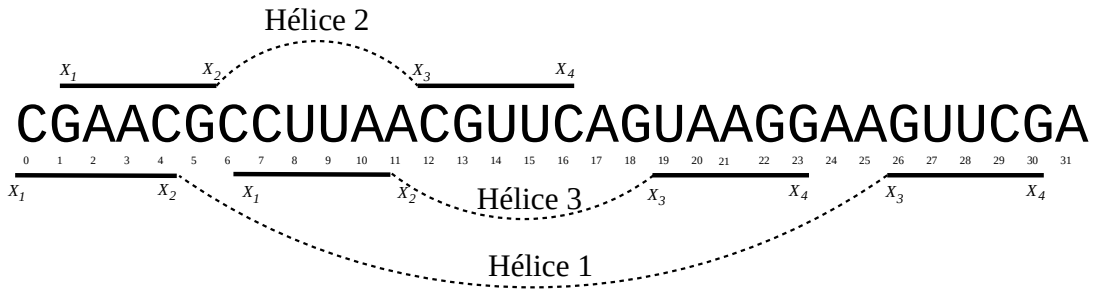


FIG. 5.14 – *(1) Description du motif recherché et (2) formalisation du motifs structuré dans le cadre CSP.*

$$
\begin{cases}
V = \{x_1, x_2, x_3, x_4\} \\
D = \begin{cases} D_1 = [1, t] \\ D_2 = [1, t] \\ D_3 = [1, t] \\ D_4 = [1, t] \end{cases} \\
Contraintes = \begin{cases} \texttt{Est\_Distant}(x_1, x_2, 5) \\ \texttt{Est\_Au\_Moins\_Distant}(x_2, x_3, 5) \\ \texttt{Est\_Au\_Moins\_Distant}(x_3, x_2, -22) \\ \texttt{Est\_Distant}(x_3, x_4, 5) \\ \texttt{Est\_Palindrome}(x_1, x_2, x_3, x_4, \mathsf{Texte}, 5, 5, 5, 22, \mathsf{complementaire}, 0, 0) \end{cases}
\end{cases}
$$

FIG. 5.15 – *Formalisation du problème.*



Formalisation

Contrainte distance (d)
Contrainte hélice (h)

$x_1 : D_1 = [0, 31]$
$x_2 : D_2 = [0, 31]$
$x_3 : D_3 = [0, 31]$
$x_4 : D_4 = [0, 31]$

Propagation

(d) & $x_2 \rightarrow [5, 31]$
(d) & $x_3 \rightarrow [10, 31]$
(d) & $x_4 \rightarrow [15, 31]$
(h) & $x_1 \rightarrow [0, 31]$ *: hélice 1*
(h) & $x_3 \rightarrow [12, 31]$ *: hélice 2*
(d) & $x_4 \rightarrow [17, 31]$
(h) & $x_2 \rightarrow [5, 31]$ *: hélice 1*
(h) & $x_4 \rightarrow [17, 31]$ *: hélice 2*
(d) & $x_1 \rightarrow [0, 26]$
(d) & $x_2 \rightarrow [5, 26]$
(d) & $x_1 \rightarrow [0, 21]$
(d) & $x_3 \rightarrow [12, 26]$
(d) & $x_2 \rightarrow [5, 21]$
(d) & $x_1 \rightarrow [0, 16]$

$x_1 = [0 - 16]$
$x_2 = [5 - 21]$
$x_3 = [12 - 26]$
$x_4 = [17 - 31]$

Fonction Retour-arrière
Instanciation $x_1 = [0, 16]$ **à 0**
(d) & $x_2 \rightarrow [5, 5]$
(h) & $x_3 \rightarrow [26, 26]$ *: hélice 1*
(d) & $x_4 \rightarrow [31, 31]$
(h) & $x_4 \rightarrow [31, 31]$ *: hélice 1*
Solution: $x_1 = 0, x_2 = 5, x_3 = 26, x_4 = 31$
Valeur suivante $x_1 \rightarrow [1, 16]$
(d) & $x_2 \rightarrow [6, 21]$
(h) & $x_1 \rightarrow [1, 16]$ *: hélice 2*
(h) & $x_3 \rightarrow [12, 26]$ *: hélice 2*
(h) & $x_4 \rightarrow [17, 31]$ *: hélice 2*
(h) & $x_2 \rightarrow [6, 21]$ *: hélice 2*
Instanciation $x_1 = [1 - 16]$ **à 1**
(d) & $x_2 \rightarrow [6, 6]$
Instanciation $x_3 = [12, 26]$ **à 12**
(d) & $x_4 \rightarrow [17, 17]$
Solution: $x_1 = 1, x_2 = 6, x_3 = 12, x_4 = 17$
Valeur suivante $x_3 \rightarrow [13, 26]$
(d) & $x_4 \rightarrow [18, 31]$
Exception lancée par (h)**&**x_4
$x_1 \rightarrow [1, 16]$**,** $x_2 \rightarrow [6, 21]$**,** $x_3 \rightarrow [12, 26]$
(d) & $x_1 \rightarrow [2, 16]$
(d) & $x_2 \rightarrow [7, 21]$
(h) & $x_1 \rightarrow [6, 16]$ *: hélice 3*
(d) & $x_2 \rightarrow [11, 21]$
(d) & $x_3 \rightarrow [16, 26]$
(d) & $x_4 \rightarrow [21, 31]$
(h) & $x_3 \rightarrow [19, 26]$ *: hélice 3*
(d) & $x_4 \rightarrow [24, 31]$
(h) & $x_3 \rightarrow [19, 26]$ *: hélice 3*
(h) & $x_4 \rightarrow [24, 26]$ *: hélice 3*
Instanciation $x_1 = [6 - 16]$ **à 6**
(d) & $x_2 \rightarrow [11, 11]$
Instanciation $x_3 = [19 - 26]$ **à 19**
(d) & $x_4 \rightarrow [24, 24]$
Solution: $x_1 = 6, x_2 = 11, x_3 = 19, x_4 = 24$
...

FIG. 5.16 – *Recherche d'un motif structuré contenant une hélice de 5 paires de nucléotides et une boucle de taille variable de 5 à 22 nucléotides.*

5.4.4 Contrainte `Est_Duplex`

`Est_Duplex`($x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, \mathsf{Texte}_1, \mathsf{Texte}_2, \mathsf{l}_{\mathsf{min}}, \mathsf{l}_{\mathsf{max}}, \mathsf{regle}, \mathsf{err}, \mathsf{type}_{\mathsf{err}}$) permet de modéliser l'occurrence d'un motif structuré mettant en jeu deux molécules indépendantes. Le motif structuré correspondant est modélisé par deux régions:

- respectivement délimitées par (x_{i_1}, x_{i_2}) sur le Texte_1 et par (x_{i_3}, x_{i_4}) sur le texte Texte_2,
- d'une longueur appartenant à l'intervalle $[\mathsf{l}_{\mathsf{min}}, \mathsf{l}_{\mathsf{max}}]$,
- et dont chaque caractère vérifie la relation P_s en utilisant la règle regle.

La modélisation d'occurrences, appartenant à deux molécules indépendantes, apporte un niveau de représentation plus complexe. Chaque molécule structurée peut être considérée comme un CSP indépendant défini par ses variables et ses contraintes jusqu'à l'application de la contrainte `Est_Duplex` reliant les deux CSP par les deux couples de variables (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}).

5.4.4.1 Fonction de propagation

Dans la contrainte `Est_Duplex`, les régions contraintes n'appartiennent pas au même texte. En conséquence, il n'est plus possible de limiter la complexité de la recherche de solutions grâce aux paramètres $\mathsf{b}_{\mathsf{min}}$ et $\mathsf{b}_{\mathsf{max}}$, comme dans la contrainte `Est_Palindrome`. La complexité de la recherche de solution par une méthode naïve redevient quadratique.

Pour palier ce problème de coût de recherche, nous utilisons une structure de données (chapitre 2, page 43), appelé *k-factor tree* [AS03]. L'arbre des suffixes (AS), correspondant à cette structure de données, permet de représenter l'ensemble des facteurs de longueur maximale $\mathsf{l}_{\mathsf{max}}$ du texte Texte_2. Sa manipulation permet de vérifier la présence d'un mot dans le texte Texte_2 en temps linéaire par rapport à la taille du mot. L'utilisation d'un AS dans le cadre de la recherche de palindrome nécessite alors:

- d'extraire un mot correspondant au premier "bras" du palindrome borné par (x_{i_1}, x_{i_2}), que nous appellerons "mot père", et de lui appliquer un traitement,
- de rechercher le mot ainsi traité dans l'AS. Les solutions obtenues lors de cette seconde étape correspondent au deuxième "bras" du palindrome borné par (x_{i_3}, x_{i_4}).

Le traitement effectué sur le "mot père" consiste à l'inverser en lui appliquant la fonction `inverse` et à remplacer chacun de ses caractères s_i par le résultat de la fonction `complement`(s_i) (par exemple pour les appariements de type Watson et Crick, `complement`(A) = U, `complement`(U) = A, `complement`(G) = C et `complement`(C) = G). L'approche avec l'AS consiste ainsi à générer *a priori* le symétrique d'un mot et de le rechercher ensuite.

Contraintes d'implémentation liées à l'utilisation d'un AS

La construction de l'AS étant effectuée définitivement à l'initialisation du CSP, la complexité de chaque recherche sera donc linéaire par rapport à la taille du mot recherché. L'utilisation d'un AS, avec l'idée d'en tirer un maximum de profit d'un point de vue efficacité, a entraîné une implémentation spécifique de la contrainte pour les deux points majeurs suivants :

- Pour la contrainte `Est_Palindrome`, la recherche d'un mot symétrique d'une région bornée par (x_{i_1}, x_{i_2}), par exemple, est couplée à l'exploration des régions bornées par (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}). Dans le cas présent, l'AS utilisé stocke l'ensemble des facteurs du

texte Texte_2 de longueur maximale l_{max}, et limite, de ce fait, la taille des mots pouvant y être recherchés à l_{max}. Il apparaît alors évident qu'effectuer une recherche de palindrome (en utilisant un AS) couplée à l'exploration de la région $[x_{i_1}, x_{i_2}]$ serait fort coûteuse. De ce fait, l'étape consistant à extraire un mot est seulement réalisée lorsqu'une des deux extrémités de ce mot devient fixe ($v_{bi} = v_{bs}$). En d'autres termes, lorsqu'une des deux variables le délimitant devient instanciée.

– A partir d'un AS, la recherche de l'ensemble des occurrences d'un mot consiste à identifier, s'il existe, le chemin d'un noeud correspondant au mot. Si ce noeud existe dans le texte, et correspond à un noeud interne, ou à une feuille, l'ensemble de ses occurrences est donné par l'ensemble de ses feuilles filles où les positions initiales sont mémorisées (Fig 2.16, section 2.3.3, page 65) . Ainsi, à partir d'un AS, on obtient pour un même coût l'ensemble des positions du mot correspondant au second "bras" du duplex.

  Ce constat nous a amené à modifier la philosophie utilisée par les fonctions précédentes de propagation de contraintes, où la fonction de propagation renvoie la première valeur d'un domaine (par rapport à l'une de ses deux bornes) respectant la $2B$-cohérence. Dans le cas présent, l'ensemble des valeurs, obtenues en un seul coup à partir de la recherche d'un mot, est mémorisé dans une liste triée. La dernière étape de la fonction de propagation consistera alors à explorer cette liste pour en déduire les bornes $2B$-cohérentes de chaque variable. Le mécanisme de propagation de cette contrainte présente un certain nombre de particularités comme nous le présentons dans la section suivante.

Besoins suscités par la modélisation de deux molécules

Un autre aspect particulier à la contrainte `Est_Duplex` concerne l'ajout d'une complexité en considèrant deux CSP "parallèles", reliés par la contrainte.

En effet, en terme biologique, la recherche d'une molécule dans un génome doit prendre en considération le fait que l'on ignore son sens de transcription. Cela revient à dire que le gène codant pour la molécule peut être localisé

– soit sur le brin d'ADN génomique $5' \rightarrow 3'$, correspond à un texte Texte tel qu'il est donné par convention;
– soit sur le brin *reverse* d'ADN génomique $3' \rightarrow 5'$, correspond à un texte Texte_{rc} traité par une inversion et de l'application de `complement` sur chacun de ses caractères .

Dans la présentation des autres contraintes, nous avons ignoré ce problème car il est facilement résolvable en effectuant à la suite deux recherches sur les textes Texte et Texte_{rc}. Dans le cas de deux textes génomiques, nous sommes obligés de le prendre en considération car chaque molécule est transcrite indépendamment de l'autre, ce qui multiplie les recherches à effectuer.

Par souci de simplification, nous avons orienté nos développements, dans un premier temps, avec l'idée que les deux molécules partageant une interaction sont de complexités inégales. L'une d'entre elles fait partie d'un CSP défini par un nombre de variables et contraintes plus important. Sans perte de généralités, nous supposons que la molécule la plus structurée est transcrite sur le brin d'ADN $5' \rightarrow 3'$.

La figure 5.17 représente un duplex formé par les deux régions bornées par (x_{i_1}, x_{i_2}) et (x_{i_3}, x_{i_4}) et schématise l'étape d'extraction et le résultat des traitements effectués sur le "mot père". En effet, pour recherche, à partir d'un même texte Texte_2, la seconde région d'un duplex

pouvant correspondre à une molécule transcrite:

- sur le brin d'ADN génomique $5' \rightarrow 3'$, il faut appliquer au mot les fonctions `inverse` et `complement` pour obtenir le mot Mot_{rc};
- sur le brin *reverse* d'ADN génomique $3' \rightarrow 5'$, il faut seulement appliquer la fonction `complement` pour obtenir le mot Mot_c.

L'étape d'extraction est effectuée seulement si une des extrémités (par exemple x_1, $v_{1_{bi}} = v_{1_{bs}}$) du mot est fixe. A partir de cette position, la longueur du mot est donnée par le minimum de $(v_{1_{bi}} + \mathsf{l_{max}})$ et $(v_{2_{bs}})$. La prise en compte de la variabilité de la longueur du mot (entre $[l_{min}, MIN(l_{max}, v_{2bs} - v_{1_{bi}})]$) est schématisée en pointillé sur la figure 5.17. Si l'on regarde les résultats des deux traitements effectué sur le "mot père", la variabilité se retrouve au début, pour le mot $\mathsf{Mot_{rc}}$, et en fin, pour le mot $\mathsf{Mot_c}$. La recherche du mot dans un AS nécessite d'en connaître le début exact afin de rechercher l'unique chemin lui correspondant à partir de la racine. Si la partie variable se trouve en fin de mot, l'ensemble des occurrences correspondant à la taille $\mathsf{l_{min}}$ du mot $\mathsf{Mot_c}$ est obtenu à partir d'un unique noeud. Par contre, si cette partie se trouve en début de mot, il est nécessaire de parcourir l'arbre pour chaque préfixe du mot $\mathsf{Mot_{rc}}$ pour obtenir l'ensemble des occurrences correspondant à la taille $\mathsf{l_{min}}$ du mot $\mathsf{Mot_{rc}}$. La recherche exhaustive est, de ce fait, moins efficace dans le second cas (exponentielle dans la taille de la zone variable) et en conséquence, nous restreignons l'utilisation de l'AS à la recherche exclusive des "mots pères" de taille minimum. Les extensions, si elles sont possibles, sont effectuées *a posteriori* par une approche naïve, qui essaie à partir de chaque occurence trouvée à partir de l'AS, d'effectuer l'extension du palindrome.

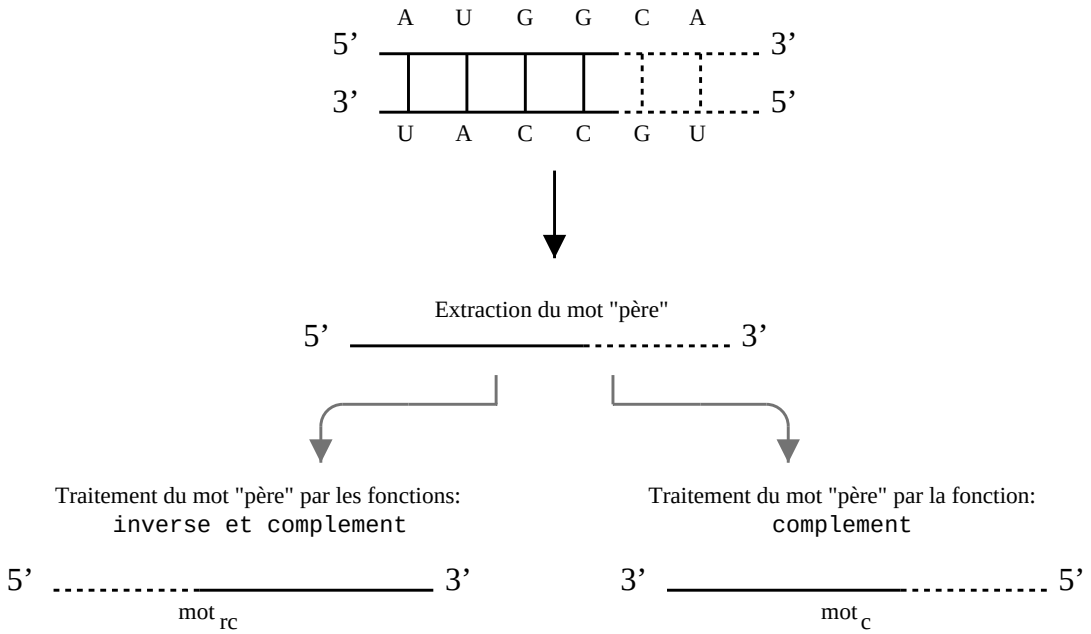


FIG. 5.17 – *Extraction du mot "père".*

Nous avons ensuite décidé d'effectuer le traitement du mot "père" non pas à partir de chaque extraction (beaucoup trop coûteux) mais une fois pour toute sur le texte utilisé pour construire l'AS. Les textes $\mathsf{Texte_{2rc}}$ et $\mathsf{Texte_{2c}}$ sont obtenus à partir du $\mathsf{Texte_2}$ en appliquant respectivement

les fonctions `inverse, complement` et la fonction `complement`. La concaténation des deux textes est ensuite utilisée pour construire l'AS$_{rc+c}$.

La fonction de propagation de la contrainte comprends les étapes suivantes:

- attente de l'instanciation de l'une des deux variables délimitant le "mot père" ;
- extraction et recherche du "mot père" dans AS$_{rc+c}$;
- mémorisation de la liste de positions dans $Liste_{x_i}$;
- parcours de la liste triée $Liste_{x_i}$ pour extraire les positions de x_3 ou x_4 (optimisé par la mémorisation d'indices croissants et décroissant), ou les positions de x_1 ou x_2 (calculées à partir de la solution (x_3, x_4) représentant le duplex de plus grande taille).

Algorithme : `Est_duplex_`x_1`(`x_1, x_2, x_3, x_4, Texte$_1$, Texte$_2$, l$_{min}$, l$_{max}$, regle`)`

$Liste_{x_1}$ `: liste de solutions`
N_{x_1} `: taille de la` $Liste_{x_1}$
x_1 `est instancié /*`$v_{1_{bi}} = v_{1_{bs}}$`*/`
Mot$_{pere}$ $\leftarrow$ Texte$_1[v_{1_{bi}}, v_{1_{bi}} + MAX(l_{min}$,$v_{2_{bi}})]$
`(`$N_{x_1}, Liste_{x_1}$`)` $\leftarrow$ `cherche_mot_dans_AS(`AS_{rc+c}`,`Mot$_{pere}$`)`
$Liste_{x_1}$ $\leftarrow$ `Etendre_Duplex(`$Liste_{x_1}$, l$_{max}$ − l$_{min}$`)`
`retourner` $Liste_{x_1}$

FIG. 5.18 – `Est_duplex_`x_1.

La fonction `Est_duplex_`x_1 (Fig 5.18) illustre le cas de figure où la recherche est amorcée à partir de l'instanciation de x_1. La liste triée de solutions $Liste_{x1}$ est obtenue pour un contexte donné (mémorisation des bornes du domaine courant de x_2). $Liste_{x1}$ est recalculée pour une nouvelle instance de x_1 ou pour un nouveau contexte.

5.4.4.2 Mécanisme de propagation

La particularité de la contrainte `Est_Duplex` est d'implémenter une méthode de propagation plus paresseuse que les précédentes. La propagation de la contrainte est suspendue tant que x_1 ou x_2 n'a pas été instanciée. A partir d'une instance de l'une ou l'autre de ces variables, une liste de valeurs compatibles pour x_3 est mémorisée. La propagation à partir du réveil de x_1 ou x_2 est effectuée sur les trois variables non instanciées. Les nouvelles valeurs à propager sont calculées à partir de la liste des occurrences dans AS_{rc+c}.

Dans l'état actuel de l'implémentation de la contrainte `Est_Duplex`, la prise en compte de la règle de comparaison ainsi que la gestion des erreurs autorisées n'ont pas encore été considérées . Ces deux améliorations nécessitent des modifications au niveau de l'algorithme de recherche de mot dans un AS.

5.5 Prototype Milpat

Les développements algorithmiques de la section précédente sont mis en oeuvre dans le prototype Milpat : **M**otifs and **I**nter-mo**L**ecular motifs searching tool using cs**P** form**A**lism and solving **T**echniques.

L'implémentation est faite avec deux objectifs : le premier est d'offrir une architecture logicielle modulaire qui accepte aisément des extensions, et le second est de mettre en oeuvre une gestion optimisée des événements de propagation. L'utilisation d'un langage orienté-objet (C++) apporte un environnement basé sur le principe d'encapsulation des objets. Ils sont ensuite manipulables uniquement au travers d'une interface. Cette définition convient parfaitement au formalisme des CSP. Cette section détaille brièvement les choix de conception qui sont fait dans MilPat.

5.5.1 Architecture

Nous introduisons le modèle entités-relations utilisé pour l'implémentation de Milpat. Le réseau et les variables et contraintes sont représentés par trois classes RÉSEAU, VARIABLE et CONTRAINTE. D'une manière générale, le réseau de contraintes est codé par des liens entre les objets VARIABLE et les objets CONTRAINTE : chaque objet CONTRAINTE contient, parmi ses champs, des références à ses variables ; chaque objet VARIABLE contient un champ indiquant la liste des contraintes dans lesquelleq elle apparaît.

Classe VARIABLE

Structure de données : Dans l'état actuel, une seule classe VARIABLE existe. Chaque variable est instanciée à partir de cette classe. La classe englobe les entiers v_{bi} et v_{bs} correspondant aux bornes du domaine, deux piles d'entiers permettant de sauvegarder les couples de valeurs définissant les bornes de la variable, ainsi qu'un champ indiquant la liste des contraintes dans lesquelles elle apparaît.

Les méthodes : Les méthodes associées à chaque instance de la classe VARIABLE sont la méthode de construction, les événements $Est \leq$ et $Est \geq$, les procédures de sauvegarde et de restauration des deux bornes de chaque variable.

Classe CONTRAINTE

Chaque type de contrainte (contraintes arithmétiques ou spécifiques) est instancié à partir d'une classe descendant de la classe CONTRAINTE.

Structure de données : La classe englobe un tableau d'entiers pour les variables sur lesquelles elle repose et un entier correspondant à son arité (la taille du tableau de variables).

Les méthodes : Plusieurs méthodes de construction sont proposées selon l'arité de la contrainte. Deux méthodes virtuelles sont données correspondant aux procédures `Revise_Borne_Inférieure` et `Revise_Borne_Supérieure`. Liées à leur caractère virtuel, elles doivent être "surchargées" pour chaque classe héritant de CONTRAINTE. La classe définit également un canevas de sauvegarde des dernières valeurs obtenues par application de la fonction de propagation.

Les classes héritant de CONTRAINTE

Chaque type de contrainte sera défini par une classe descendant de la classe CONTRAINTE. Chacune utilise le constructeur de CONTRAINTE approprié selon son arité et fournit sa propre implémentation des méthodes de propagation et des procédures de sauvegarde des dernières solutions obtenues (section 5.4, page 161). Le nombre de champs prévu pour le stockage de l'information est multiplié par l'arité de la contrainte (dans le cas de la contrainte `Est_Palindrome`, quatre groupes de valeurs sont mémorisés). Dans le cas de la contrainte `Est_Duplex`, deux listes statiques de mémorisation de valeurs sont nécessaires ainsi qu'un jeu d'indices permettant de parcourir rapidement ces listes. L'étape principale de l'algorithme de propagation consiste à aller réveiller toutes les contraintes liées à une variable chaque fois que le domaine de cette variable est modifié.

Classe RÉSEAU

Le réseau de contraintes englobe également les références objets-variables et les objets-contraintes.

Structure de données : Des entiers comptabilisent le nombre total de variables et contraintes et deux champs sont prévus pour référencer les variables et contraintes via des tableaux.

Les méthodes : Les principales méthodes sont les procédures pour l'instanciation des variables du réseau, l'ensemble des méthodes d'ajout de contraintes et les procédures de propagation initiale des contraintes, de `retour-arrière` et d'impression des solutions du CSP.

Remarques

L'architecture de MilPat fournit un canevas pour assembler des contraintes hétérogènes de manière uniforme. Le code peut être ainsi organisé de manière modulaire puisque non seulement la propagation d'une contrainte ne nécessite d'information que sur les états de ses propres variables et n'a d'effet que sur elles, et que, d'autre part, toutes les contraintes attachées à une variable peuvent être réveillées indépendamment, l'une après l'autre, dans n'importe quel ordre. Cet aspect modulaire ou cadre intégrateur, dans le cas des contraintes, permet ainsi d'exploiter et de combiner des algorithmes efficaces encapsulés dans des classes héritant de CONTRAINTE. A partir de cette architecture générique, le nombre de classes implémentant des contraintes peut être étendu simplement.

5.5.2 Résultats

Cette section s'inscrit dans la continuité du chapitre 3 où nous avons présenté une étude détaillée d'un ensemble de logiciels généralistes. Nous présentons les résultats de recherche de deux motifs structurés : ARNt et snoARN. Les jeux de données sont identiques à ceux utilisés dans le chapitre 3.

5.5.2.1 Recherche d'ARN de transfert

Nous avons écrit un CSP à partir du descripteur d_1 (section 3.4.3.1, page 92) pour rechercher les ARNt sur le génome de *E.coli* et à partir du descripteur d_2 pour le génome de *S. cerevisiae*. Le CSP dans le premier cas a nécessité 15 variables et 21 contraintes (14 contraintes `Est_Au_Moins_Distant` , 4 contraintes `Est_Palindrome` et 3 contraintes `Est_De_Nature`) et dans le second cas 15 variables et 20 contraintes. La figure 5.19 schématise la structure secondaire de l'ARNt à partir de laquelle le CSP a été modélisé (Fig 5.20). La figure 5.21 donne le programme CSP correspondant au descripteur d_1.

Les résultats (Table 5.1) obtenus avec le prototype MilPat, comparés aux résultats du chapitre 3, sont très encourageants. En effet, nous obtenons le même nombre de solutions pour des temps de calculs similaires à ceux de Patscan, plus efficaces en terme de temps d'exécution que RnaMot ou RnaMotif. La version actuelle montre déjà une efficacité très intéressante qui ne peut être qu'améliorée en utilisant un ordre dynamique de propagation des contraintes. Les améliorations à apporter à notre prototype concernent d'une part des optimisations d'efficacité et d'autre part la possibilité d'utiliser une fonction de score à l'instar de RnaMotif ou Palingol pour ainsi améliorer la spécificité des résultats (en contrôlant par exemple le nombre global de mésappariement etc...)

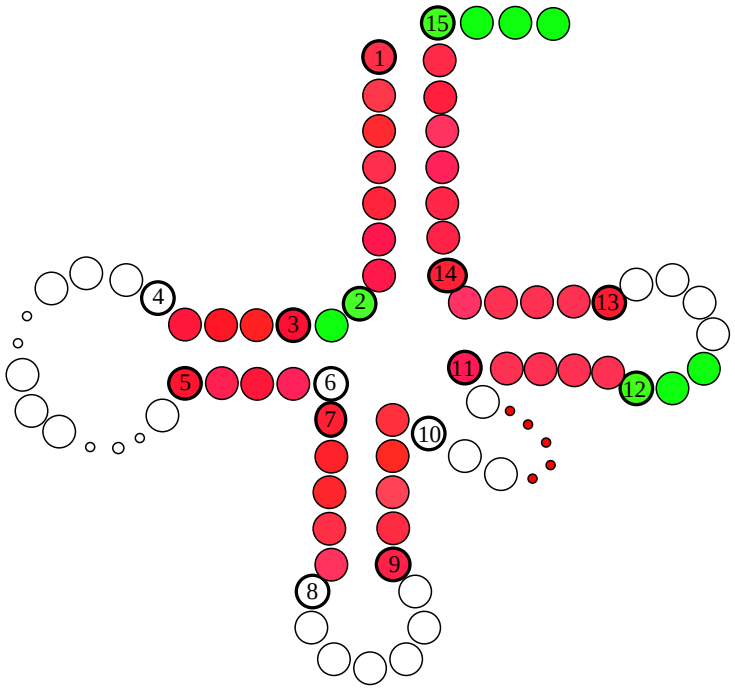


FIG. 5.19 – *La structure secondaire de l'ARNt modélisé par d_1. Les positions recherchées par les variables du CSP sont représentées par l'indice de la variable entourée d'un cercle.*

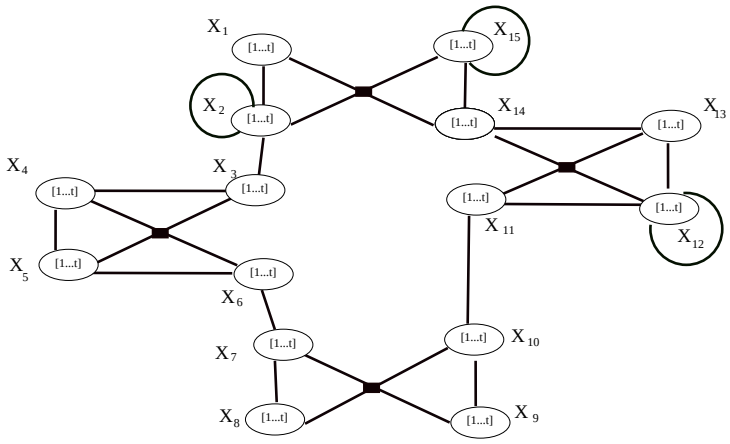


FIG. 5.20 – *Représentation en réseau du CSP. Chaque variable du problème est représentée par un cercle où son domaine de valeurs est donné, chaque contrainte* `Est_Au_Moins_Distant` *est représentée par un trait, chaque contrainte* `Est_De_Nature` *par un arc de cercle posé sur le domaine de la variable, et chaque contrainte* `Est_Palindrome` *par quatre traits reliant les quatre variables à un carré.*

Les variables

variable *x_1 = MyNet->Makevariable($''x''_1, 0, TheSeq->SeqLen$);
...
variable *x_{15} = MyNet->Makevariable($''x''_{15}, 0, TheSeq->SeqLen$);

Les contraintes

MyNet->AddConstraintEqDelta($x_1, x_2, 7$);
MyNet->AddConstraintEqDelta($x_2, x_3, 2$);
MyNet->AddConstraintIneqDelta($x_3, x_4, 3, -4$);
MyNet->AddConstraintIneqDelta($x_4, x_5, 4, -14$);
MyNet->AddConstraintIneqDelta($x_5, x_6, 3, -4$);
MyNet->AddConstraintEqDelta($x_6, x_7, 1$);
MyNet->AddConstraintEqDelta($x_7, x_8, 5$);
MyNet->AddConstraintIneqDelta($x_8, x_9, 6, -60$);
MyNet->AddConstraintEqDelta($x_9, x_{10}, 5$);
MyNet->AddConstraintIneqDelta($x_{10}, x_{11}, 2, -22$);
MyNet->AddConstraintEqDelta($x_{11}, x_{12}, 5$);
MyNet->AddConstraintIneqDelta($x_{12}, x_{13}, 3, -7$);
MyNet->AddConstraintEqDelta($x_{13}, x_{14}, 5$);
MyNet->AddConstraintEqDelta($x_{14}, x_{15}, 7$);
MyNet->AddConstraintHelix($x_1, x_{14}, x_2, x_{15}, TheSeq, 7, 7, 44, 134, 2$);
MyNet->AddConstraintHelix($x_3, x_5, x_4, x_6, TheSeq, 3, 4, 4, 14, 1$);
MyNet->AddConstraintHelix($x_7, x_9, x_8, x_{10}, TheSeq, 5, 5, 6, 60, 1$);
MyNet->AddConstraintHelix($x_{11}, x_{13}, x_{12}, x_{14}, TheSeq, 5, 5, 3, 7, 1$);
MyNet->AddConstraintMotif($x_2, TheSeq, "U"$);
MyNet->AddConstraintMotif($x_{12}, TheSeq, "NUC"$);
MyNet->AddConstraintMotif($x_{15}, TheSeq, "CCA"$);

FIG. 5.21 – *Programme CSP (équivalent au descripteur d_1) pour rechercher un ARNt.*

Programmes	Solutions *unique*	Solutions *double*	Vrai Positifs	Faux Négatifs	Faux Positifs	Temps
(A) : Génome *E.coli* - Descripteur d_1						
PatScan avec chev.	545					1 min. 32
RnaMot/RnaMotif	*238*	*307*	87	1	458	4 s. et 2 min.
MilPat	545					39 s
	238	*307*	87	1	458	
(B) : Génome *S. cerevisiae* - Descripteur d_2						
PatScan avec chev.	7313791					1h40
RnaMotif	*849979*	*6463812*	274	1	7313517	8h40
RnaMot	7313806					
	849982	*6463824*	274	1	7313532	92 h.
MilPat	7313791					1h52
	849979	*6463812*	274	1	7313517	

TAB. 5.1 – *Résultats obtenus avec les logiciels généralistes Patscan, RnaMotif, RnaMot et MilPat (ordre de propagation des contraintes non optimisé) en recherchant les ARN de transfert (A) : dans le génome de E.coli et (B) : dans le génome de Saccharomyces cerevisiae.*

5.5.2.2 Recherche de snoARN à boîtes C/D

Nous avons effectué deux types de recherche pour les snoARN à boîte C/D sur les génomes de *Pyroccocus abyssi* et *Pyrobaculm aerophilum*:

- nous avons modélisé par deux variables les deux boîtes C et D séparées par une distance appartenant à l'intervalle $[38, 67]$.
- nous avons modélisé l'interaction entre le candidat snoARN et l'ensemble du génome par une contrainte duplex (Fig 5.22). La représentation en réseau du CSP 5.23 obtenu comprend alors 5 variables, 7 contraintes (4 contraintes `Est_Au_Moins_Distant` , 1 contrainte `Est_Duplex` et 2 contraintes `Contient`). La figure 5.24 représente le programme CSP obtenu.

Les résultats obtenus n'ont pas été comparés à ceux du chapitre 3 où nous la recherche de snoARN avec duplex a été effectuée par deux utilisations simultanées de Patscan. La recherche d'interactions était exclusivement faite entre les candidats, obtenus par la première recherche de snoARN seul (avec la prise en compte d'une erreur pour chaque boîte), et l'ensemble des ARNt et ARNr annotés du génome correspondant. Le duplex étant modélisé par un élément de référence hélice, il n'était pas possible avec PatScan d'ajouter à chaque candidat l'ensemble du génome (lors de la construction de la base de données *ad hoc*).

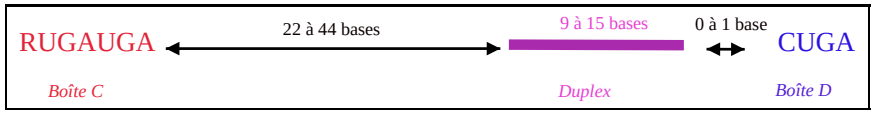


FIG. 5.22 – *Modélisation du snoARN et de son interaction.*

Dans la cas présent, la recherche d'interactions peut être effectuée sur l'ensemble d'un génome. Cependant, la prise en compte d'erreurs par la contrainte `Est_Duplex` n'est pas encore opérationnelle. On observe (Tab 5.2) une faible spécificité lors de la recherche de snoARN seul.

Ce résultat est lié à la non prise en compte d'erreurs au niveau des deux boîtes. Il est intéressant de noter cependant que la recherche d'interactions avec une autre région du génome s'effectue en un temps très convenable, et que malgré l'impossibilité d'autoriser la recherche de duplex avec erreurs, le nombre de candidats (correspondant aux solutions "uniques") présentant une à plusieurs interactions est obtenu en un temps de calcul très intéressant.

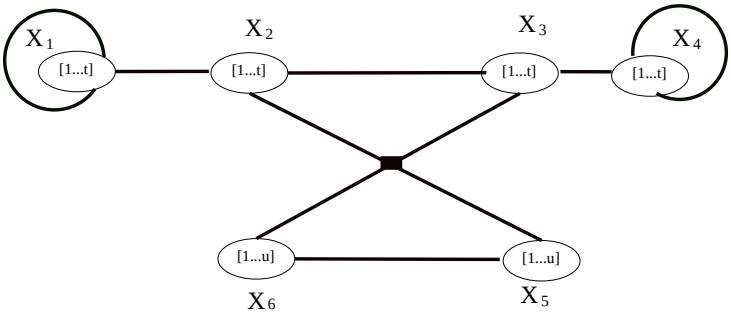


FIG. 5.23 – *Représentation en réseau du CSP. Chaque variable du problème est représentée par un cercle où son domaine de valeurs est donné, chaque contrainte* `Est_Au_Moins_Distant` *est représentée par un trait, chaque contrainte* `Est_De_Nature` *par un arc de cercle posé sur le domaine de la variable, et la contrainte* `Est_Duplex` *par quatre traits reliant les quatre variables à un carré.*

Les variables

variable *x_1 = MyNet->Makevariable($''x''_1, 0, TheSeq->SeqLen$);
variable *x_2 = MyNet->Makevariable($''x''_2, 0, TheSeq->SeqLen$);
variable *x_3 = MyNet->Makevariable($''x''_3, 0, TheSeq->SeqLen$);
variable *x_4 = MyNet->Makevariable($''x''_4, 0, TheSeq->SeqLen$);
variable *x_5 = MyNet->Makevariable($''x''_5, 0, TheSeq->SeqLen$);
variable *x_6 = MyNet->Makevariable($''x''_6, 0, TheSeq->SeqLen$);

Les contraintes

MyNet->AddConstraintIneqDelta($x_1, x_2, 29, -51$);
MyNet->AddConstraintIneqDelta($x_2, x_3, 9, -15$);
MyNet->AddConstraintIneqDelta($x_3, x_4, 0, -1$);
MyNet->AddConstraintIneqDelta($x_5, x_6, 9, -15$);
MyNet->AddConstraintMotif($x_1, TheSeq, "RUGAUGA"$);
MyNet->AddConstraintMotif($x_4, TheSeq, "CUGA"$);
MyNet->AddConstraintDuplex($x_2, x_3, x_5, x_6, TheSeq, TheSeq, root, 9, 15, 0$);

FIG. 5.24 – *Programme CSP pour rechercher un snoARN partageant une interaction.*

	Solutions		**Vrai**	**Faux**	**Faux**	**Temps**
	unique	*double*	**positifs**	**négatifs**	**positifs**	
Génome *Pyrococcus abyssi*						
snoARN seul	81		39	20	42	2 s
	81	*0*				
snoARN + duplex	1708		23	36	1685	2 min.
	65	*1707*				
Génome *Pyrobaculum aerophilum*						
snoARN seul	52		30	21	22	2 s
	52	*0*				
snoARN + duplex	1254		30	21	1224	2 min.
	51	*1203*				

TAB. 5.2 – *Résultats obtenus par MilPat avec les génomes de Pyrococcus abyssi et Pyrobaculum aerophilum en recherchant un snoARN seul et un snoARN avec une interaction avec l'ensemble du génome.*

5.5.3 Discussion

Le premier objectif était de pouvoir représenter l'ensemble des éléments définis dans le chapitre 3 dans le cadre d'un formalisme CSP. La recherche d'ARNt illustre ce premier point atteint. La spécification d'éléments de type mots et hélices se fait de façon souple au moyen de variables et de contraintes. De même, MilPat permet d'appréhender de façon similaire plusieurs niveaux de complexité de motifs structurés:

- les répétitions entre mots sont gérées de la même façon que les palindromes,
- la recherche de pseudo-noeuds, de triples hélices ou quadruples hélices se spécifie par les combinaisons de variables et des deux contraintes `Est_Palindrome` et `Est_Répétition`.

On remarque de plus que MilPat effectue des recherches en des temps d'exécution raisonnable en comparaison des autres logiciels en se plaçant, pour un même descripteur, au niveau du meilleur d'entre eux pour la recherche d'ARNt. L'implémentation d'heuristiques dans le prototype actuel laisse présager des améliorations notables de son efficacité.

Le second objectif était de pouvoir prendre en considération l'interaction entre deux molécules indépendantes. Le formalisme CSP se prête bien à la modélisation d'un tel problème et l'utilisation d'une structure de données pour représenter le texte où la recherche de duplex est effectuée sont illustrés par l'exemple de la recherche de snoARN. Les premiers résultats sont encourageants, et positionnent déjà MilPat dans une catégorie à part grâce à la possibilité de formaliser les interactions inter-moléculaires. L'exemple que nous avons proposé part d'un modèle simple de motifs structurés, composé d'une première molécule définie par plusieurs variables et d'une seconde définie seulement par deux variables (le duplex). Il est cependant déjà possible de modéliser des molécules de complexité plus importante et partageant une interaction.

Conclusion - Perspectives

Cette thèse avait pour objectif d'étudier et de développer des approches de recherche de motifs structurés. Dans cette perspective, nous avons développé le prototype MilPat (**M**otifs and **I**nter-mo**L**ecular motifs searching tool using cs**P** form**A**lism and solving **T**echniques) . Nous nous sommes intéressés à la problématique de la recherche de motifs en nous appuyant sur les formalismes et outils existant et en essayant d'intégrer des spécifications nous permettant d'aller plus loin dans cette recherche. Nous avons présenté un prototype générique dédié à la représentation et à la résolution de problèmes de recherche de motifs structurés, basé sur les notions de contraintes et d'objets.

Analyse de l'existant

Nous avons réalisé une étude comparative des outils existants, qu'ils soient généralistes ou spécifiques, en recherchant deux types de motifs structurés: les petits ARN nucléolaires (snoRNAs) à boîtes C/D et les ARNs de transfert (tRNAs). Nous avons utilisé des jeux de données de type banque de données génomique et banque de données spécialisée, pour évaluer la sensibilité et la spécificité des différents outils.

Une recherche efficace de motifs structurés est conditionnée par la connaissance des éléments caractéristiques du motif recherché, quel que soit le logiciel utilisé. Pour les deux groupes de logiciels, la recherche des ARNt a donné de très bons résultats comparé à la recherche des snoRNA. Ce résultat est très certainement lié au fait que les ARN de transfert constituent une famille homogène et relativement bien structurée (éléments de structure, bases (semi)invariantes...).

Dans le cas des snoARN, les outils généralistes ne permettent de spécifier ni les relations intermoléculaires ni la présence d'éléments optionnels. De ce fait ni l'existence du duplex formé entre le petit ARN et sa cible ni la présence optionnelle d'une hélice terminale ne peuvent être utilisées. En tenant compte uniquement des éléments de séquences, nous avons obtenu un nombre important de faux positifs et de faux négatifs. A partir des conclusions de cette étude, nous avons défini les éléments indispensables permettant de caractériser un motif puis les nouveaux éléments que nous souhaitions intégrer à notre logiciel.

MilPat

Le second objectif du travail de thèse a porté sur la proposition d'un modèle et les développements informatiques d'un outil permettant de rechercher dans les séquences génomiques des régions non codantes fonctionnelles pouvant mettre en jeu des interactions internes à la

molécule aussi bien qu'avec d'autres régions d'un génome cible. Cette recherche s'effectue sur la base d'une signature caractérisant les structures des régions fonctionnelles, ses interactions avec d'autres séquences et les positionnements relatifs de ces éléments. Dans cette perspective, nous avons développé MilPat. L'analyse comparative de séquences est une approche très efficace pour la détermination des structures secondaires et de certaines interactions tertiaires des molécules d'ARN. Les changements de bases compensatoires (Fig. 1.22) observés entre plusieurs séquences sont à la base de la prédiction des structures secondaires par les méthodes phylogénétiques. Ce phénomène s'observe au cours de l'évolution des molécules : quand une base impliquée dans un appariement mute, la base complémentaire mute également pour préserver l'appariement et donc la structure secondaire. Le prototype est intéressant au regard des possibilités qu'il offre déjà, et des perspectives qu'il permet.

L'architecture de Milpat présente l'avantage d'être modulaire et générique et permet ainsi d'accepter de nombreuses extensions dont nous faisons une liste non exhaustive pour conclure.

Perspectives à court terme

Interface utilisateur

Un des points les plus importants concernant les perspectives de MilPat est de lui adjoindre une interface utilisateur sous la forme d'un langage. A l'heure actuelle, un descripteur correspond à un programme C++ (deux exemples sont donnés pour la recherche d'ARNt Fig 5.21 et de snoARN Fig 5.24). Au cours du chapitre 3, nous nous sommes intéressés aux possibilités offertes par les langages d'un certain nombre de logiciels génériques de recherche de motifs structurés. Ce premier travail de comparaison, lié aux possibilités et pouvoirs d'expression offerts par chaque langage, nous a permis de mettre en relief les objectifs pour définir un langage permettant de décrire l'ensemble des contraintes via le langage de Milpat. Ce futur travail nécessite plusieurs niveaux de développement.

Description d'un langage Cette étape consistera à entamer une réflexion sur le niveau d'expressivité que nous souhaitons donner au langage de Milpat. Nous voulons trouver un juste compromis entre la richesse d'expression déclarative de Palingol et la simplicité d'utilisation du langage de RnaMot. RnaMotif est aussi un excellent point de départ de notre réflexion. Le résultat de cette première étape décrira le langage offert à l'utilisateur.

Analyseur lexical du langage A partir de l'énumération des règles de représentation du langage, l'étape suivante sera de mettre au point un analyseur lexical. Ce programme, étant donné un flot d'entrée, a pour but de reconnaître les expressions spécifiées par l'utilisateur et de les partitionner.

Analyseur syntaxique du langage Une fois l'analyse lexicale effectuée, l'analyse syntaxique permet, étant donnée la grammaire décrivant le langage, de vérifier qu'un texte de spécification est conforme à la grammaire et dans ce cas de construire l'arbre syntaxique correspondant.

Ajout de contraintes

Un autre niveau d'améliorations peut être apporté à MilPat en étendant son pouvoir d'expression par le biais de nouvelles contraintes. Nous avons précédemment décrit l'interface des contraintes `Contient` et `Est_Composé`, lesquelles permettent de définir un mot sous forme de tables de fréquences ou de probabilités. L'intégration de ces contraintes au prototype MilPat peut se faire aisément par l'implémentation d'algorithmes exploitant des tables de fréquences [Sto90].

Motifs optionnels et fonction de score

Les snoARN, dont nous avons décrit les différents éléments caractéristiques de la famille, présentent des éléments optionnels : les boites C' et D' et une hélice terminale. Ces éléments sont absents ou trop dégénérés chez certaines molécules. Il est alors intéressant de pouvoir les spécifier par le biais de variables et contraintes dont la violation n'entraîne par l'arrêt de la recherche.

L'implémentation de telles contraintes pourra être couplée à la mise au point d'une fonction de score permettant l'adjonction d'une valeur mesurant le poids des contraintes violées. Des extensions du formalisme CSP permettent d'évaluer une solution potentielle en terme de coût, de possibilité, de priorité ou de probabilité. Dans toutes ces approches, les contraintes, qu'elles soient "dures" ou préférablement satisfaites, sont exprimées sous la forme de contraintes classiques, podérées par un degré de préférence ou d'incertitude [SFV95].

Perspectives à moyen terme

L'ordre de recherche

L'ordre dans lequel sont recherchés les différents éléments composant le motif structuré influe considérablement sur l'efficacité de la recherche. En effet, si un élément est jugé plus descriminant qu'un autre, il sera plus intéressant de le rechercher en premier. Une première approche par les CSP [Dec03] permet de calculer un ordre optimisé statique ou dynamique d'instanciation des variables en fonction du nombre ou de la nature des contraintes portant sur elle.

Une autre approche est proposée par Myers [MM93b]. Elle utilise le critère de fréquence d'un élément donné par un tirage aléatoire d'un élément du motif structuré et une recherche du nombre d'occurrences le représentant. Cette simulation est coûteuse en temps.

Une autre approche, proposée par Gras [Gra97], intègre une estimation analytique du critère de choix. Le principe repose sur les hypothèses :

- moins un élément a d'occurrences et plus l'élagage de l'arbre sera efficace,
- plus la description d'un élément sera représentée par différentes possibilités (lié au niveau de dégénérescence de la description) et plus la recherche dans l'arbre sera lente.

Il apparaît donc intéressant de trouver un critère pour déterminer la potentialité d'un élément en terme de nombre d'occurrences et de sous-éléments représentés.

RnaMot [GMC90] intègre une méthode de calcul d'un ordre de recherche préférentiel, fonction également de l'analyse des éléments du descripteur.

Motifs alternatifs

La structure secondaire d'un ARN n'est pas statique, une molécule d'ARN étant généralement capable de réarrangements conformationnels. La conformation structurale adoptée par un ARN à un instant donné est modulée par l'interaction avec de nombreux ligands. Des exemples du chapitre I illustrent cette possibilité:

– le cas de l'`ARN-OUT` (chapitre 1, section 1.3.3.2, page 31) dont la structure secondaire individuelle est une tige-boucle et qui est capable de s'apparier avec l'`ARN-IN`
– le cas de l'`ARN-III` (chapitre 1, section 1.3.3.3, page 32) dont la structure secondaire est composée de 14 tige-boucles; cet ARN présente deux domaines d'interactions avec deux autres ARN nécessitant l'ouverture de sa structure secondaire pour s'apparier avec l'un ou l'autre de ces cibles.

Ainsi, l'intégration de la riche combinatoire d'interactions intra ou intermoléculaires d'un ARN dans Milpat serait un travail original en soi. Une piste serait de s'intéresser aux CSP dynamiques [MF90, Dec03], lesquels sont particulièrement adaptés à des applications où les informations descriptives du problème sont sujettes à des modifications observées (l'environnement du problème évolue) pour explorer certaines configurations.

Bibliographie

[AGM+90] SF Altschul, W Gish, W Miller, EW Myers, and DJ Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, 1990.

[AHV+01] L Argaman, R Hershberg, J Vogel, G Bejerano, EG Wagner, H Margalit, and S Altuvia. Novel small rna-encoding genes in the intergenic regions of escherichia coli. *Curr Biol*, 11(12):941–50, 2001.

[ALM+00] PB Arimondo, L Lacroix, M Mills, H Klump, JC François, J Lacoste, and JL Mergny. Conformations inhabituelles de l'adn. *Regards sur la Biochimie*, 2:25–37, 2000.

[Amb01] V Ambros. micrornas: tiny regulators with great potential. *Cell*, 107(7):823–6, 2001.

[AOK02] I Abouelhoda, M, E Ohlebush, and S Kurtz. Optimal exact string matching based on sux arrays. In *Proc. Int Symp. on String Processing and Information Retrieval*, pages 31–43, 2002.

[AS03] J. Allali and M. F. Sagot. The at most k-deep factor tree. *Theory of Computer Science*, 2003. in submission.

[AWFZ+97] S Altuvia, D Weinstein-Fischer, A Zhang, L Postow, and G Storz. A small stable RNA induced by oxidative stress: role as a pleiotropic regulator and antimutator. *Cell.*, 90(1):43–53, 1997.

[AWN94] RB Altman, B Weiser, and HF Noller. Constraint satisfaction techniques for modeling large complexes: Application to the central domain of 16s ribosomal rna. In *Proceedings of the second international conference on Intelligent Systems for Molecular Biology*, pages 10–18, 1994.

[Bau96] DC Baulcombe. Rna as a target and an initiator of post-transcriptional gene silencing in transgenic plants. *Plant Mol. Biol.*, 3:79–88, (1996).

[BC56] J Brachet and H Chatrenne. The function of the nucleus in the synthesis of cytoplasmic proteins. *Cold Spring Harb Symp Quant Biol*, 21:329–37, 1956.

[BC97] JP Bachellerie and J Cavaille. Guiding ribose methylation of rrna. *Trends Biochem Sci*, 22(7):257–61, 1997.

[BCH02] JP Bachellerie, J Cavaille, and A Huttenhofer. The expanding snorna world. *Biochimie*, 84(8):775–90, 2002.

[BCHH01] E Bernstein, AA Caudy, SM Hammond, and GJ Hannon. Role for a bidentate ribonuclease in the initiation step of RNA interference. *Nature*, 409(6818):363–6, 2001.

[BCL+01] W Brown, J, P Clark, G, J Leader, D, G Simpson, C, and T Lowe. Multiple snorna gene clusters from arabidopsis. *RNA.*, 7(12):1817–1832, 2001.

[BD99] Soldano H. Bouthinon D. A new method to predict the consensus secondary structure of a set of unaligned RNA sequences. *Bioinformatics*, 15(10):785–798, 1999.

[BDBH99] J Baeyens, K, L De Bondt, H, and R Holbrook. Structure of an RNA double helix including uracil-uracil base pairs in an internal loop. *Nat. Struct. Biol.*, 2:56–62, 199.

[BFPP99] V Bourdeau, G Ferbeyre, M Pageau, and R Paquin, B andCedergren. The distribution of RNA motifs in natural sequences. *Nucleic Acids Res*, 27(22):4457–67, 1999.

[BHSR03] J Brennecke, DR Hipfner, A Stark, and and Cohen SM Russell, RB. bantam encodes a developmentally regulated microrna that controls cell proliferation and regulates the proapoptotic gene hid in drosophila. *Cell*, 113(1):25–36, 2003.

[BK99] P Barahona and L Krippahl. Applying constraint programming to protein structure determination. *CP*, pages 289–302, 1999.

[BKR+00] Y Benito, FA Kolb, P Romby, G Lina, J Etienne, and F Vandenesch. Probing the structure of rnaiii, the staphylococcus aureus agr regulatory rna, and identification of the rna domain involved in repression of protein a expression. *RNA*, 6(5):668–79, 2000.

[BKV96] B Billoud, M Kontic, and A Viari. Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. *Nucleic Acids Res*, 24(8):1395–403, 1996.

[Bla00] EH Blackburn. The end of the (DNA) line. *Nat Struct Biol.*, 7(10):847–850, 2000.

[BM77] RS Boyer and JS Moore. A fast string searching algorithm. In *Communications of the ACM*, volume 20, pages 762–772, 1977.

[BMN+95] JP Bachellerie, B Michot, LM Nicoloso, A Balakin, J NI, and J Fournier, M. Antisens snornas : a family of nucleolar rnas with long complementarities to rrna. *Trends Biochem. Sci.*, 20:261–264, 1995.

[BO97] F Benhamou and W Older. Applying interval arithmetic to real, integer and boolean constraint. *J. Logic Programming*, 31(1):1–24, 1997.

[BPB+97] FR Blattner, G Plunkett, CA Bloch, NT Perna, V Burland, M Riley, J Collado-Vides, JD Glasner, CK Rode, GF Mayhew, J Gregor, HA Davis, NW Kirkpatrick, MA Goeden, DJ Rose, B Mau, and Y Shao. The complete genome sequence of escherichia coli k-12. *Science,*, 277:1453–1474, 1997.

[Bra98] D Branch, A. A good antisense molecule is hard to find. *Trends Biochem Sci.*, 23(2):45–50, 1998.

[BSF96] G Balakin, A, L Smith, and J Fournier, M. The RNA world of the nucleolus: two major families of small rnas defined by different box elements with related functions. *Cell*, 86(5):823–834, 1996.

[Buc03] S Buckingham. The major world of micrornas. In *The 2nd Horizon Symposium*. Nature Publishing Group and Aventis, 2003.

[BYG92] R Baeza-Yaltes and GH Gonnet. A new approach to text searching. In *Communications of the ACM*, volume 35, pages 74–82, 1992.

[BYP92] R Baeza-Yaltes and H Perleberg, C. Fast and practical approximate string matching. In *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, volume 644, pages 185 – 192. Springer-Verlag, 1992.

[Car02] GG Carmichael. Medicine: silencing viruses with rna. *Nature*, 418(6896):379–80, 2002.

[CB96] J Cavaille and JP Bachellerie. Processing of fibrillarin-associated snornas from pre-mrna introns: an exonucleolytic process exclusively directed by the common stem-box terminal structure. *Biochimie*, 78(6):443–56, 1996.

[CB98] J Cavaille and JP Bachellerie. Snorna-guided ribose methylation of rrna: structural features of the guide RNA duplex influencing the extent of the reaction. *Nucleic Acids Res*, 26(7):1576–87, 1998.

[CB03] J Cavaille and JP Bachellerie. De nouveaux gènes : les ARN non messagers. *Pour la science*, 306:82–89, Avril 2003.

[CBF+03] G Cohen, V Barbe, D Flament, M Galperin, R Heilig, R Ripp, O Lecompte, D Prieur, O Poch, J Quellerou, J Thierry, JC andVan der Oost, J Weissenbach, Y Zivanovic, and P Forterre. An integrated analysis of the genome of the hyperthermophilic archaeon pyrococcus abyssi. *Mol. Mircrobiol*, 6(47):1495,1512, 2003.

[CBK+00] J Cavaille, K Buiting, M Kiefmann, M Lalande, CI Brannan, B Horsthemke, JP Bachellerie, J Brosius, and A Huttenhofer. Identification of brain-specific and imprinted small nucleolar RNA genes exhibiting an unusual genomic organization. *Proc Natl Acad Sci U S A*, 97(26):14311–6, 2000.

[CDH01] RJ Carter, I Dubchak, and SR Holbrook. A computational approach to identify genes for functional rnas in genomic sequences. *Nucleic Acids Res*, 29(19):3928–38, 2001.

[CH01] M Crochemore and C Hancart. *Algorithmique du texte*. Vuibert Informatique, 2001.

[Che99] F Chetouani. *Développement d'un logiciel d'édition/analyse de la structure secondaire des ARN - Recherche informatique de motifs ARN structurés et d'enzymes de modifications de nucléotides*. PhD thesis, Université Paul-Sabatier Toulouse III, 1999.

[Cho57] N Chomsky. Syntactic structures. In *The Hague: Mouton*, 1957.

[CLH+02] S Chen, EA Lesnik, TA Hall, R Sampath, RH Griffey, DJ Ecker, and LB Blyn. A bioinformatics based approach to discover small rna genes in the escherichia coli genome. *Biosystems*, 65(2-3):157–77, 2002.

[Cor88] F Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Res.*, 16(22):10881–10890, 1988.

[CR94] M Crochemore and W Rytter. *Text algorithms*. Oxford University Press, 1994.

[CT91] M Chastain and J Tinoco, I. Structural elements in rna. *Nucleic Acids Res. Mol. Biol.*, 41:131–177, 1991.

[dBT90] Y d'Aubenton, Carafa, E Brody, and C Thermes. Prediction of rho-independent escherichia coli transcription terminators. a statistical analysis of their RNA stem-loop structures. *J Mol Biol*, 216(4):835–58, 1990.

[DC00] M Daniel, R and A Cowan, D. Biomolecular stability and life at high temperatures. *Cell Mol Life Sci.*, 57(2):250–264, 2000.

[Dec03] R Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[Del03] F Delaplace. Analyse statique : du Calcul Haute Performance à la Bioinformatique. HDR : Habilitation à diriger des recherches, Novembre 2003.

[DLO97] M Dsouza, N Larsen, and R Overbeek. Searching for patterns in genomic data. *Trends Genet*, 13(12):497–8, 1997.

[DOL01] P Dennis, A Omer, and T Lowe. A guided tour: small RNA function in archaea. *Molecular Microbiology*, 40(3):509, 2001.

[Doo99] WF Doolittle. Phylogenetic classification and the universal tree. *Science*, 284:2124–2149, 1999.

[DPS03] JG Doench, CP Petersen, and PA Sharp. sirnas can function as mirnas. *Genes Dev*, 17(4):438–42, 2003.

[DS94] S Dong and DB Searls. Gene structure prediction by linguistic methods. *Genomics*, 23(3):540–51, 1994.

[ED94] SR Eddy and R Durbin. Rna sequence analysis using covariance models. *Nucleic Acids Res*, 22(11):2079–88, 1994.

[Edd96] SR Eddy. Rnabob: a program to search for rna secondary structure motifs in sequence databases. Manual, 1996.

[Edd99] SR Eddy. Noncoding rna genes. *Current Opinion in Genetics and Developement,*, 1999.

[Edd01] SR Eddy. Non-coding rna genes and the modern rna world. *Nat Rev Genet*, 2(12):919–29, 2001.

[EGJR01] I Eidhammer, D Gilbert, I Jonassen, and M Ratnayake. A constraint based structure description language for biosequences. *Constraints*, 6:173–200, 2001.

[EKW+00] D Ermolaeva, M, G Khalak, H, O White, O Smith, H, and L Salzberg, S. Prediction of transcription terminators in bacterial genomes. *J Mol Biol.*, 4(1):27–33, 2000.

[ELT01] SM Elbashir, W Lendeckel, and T Tuschl. RNA interference is mediated by 21- and 22-nucleotide rnas. *Genes Dev*, 15(2):188–200, 2001.

[EM96] N El Mabrouk. *Rechercher approchée de motifs. Application à des séquences biologiques structurées.* PhD thesis, Université de Paris VII, 1996.

[EML96] N El-Mabrouk and F Lisacek. Very fast identification of RNA motifs in genomic DNA. application to trna search in the yeast genome. *J Mol Biol*, 264(1):46–55, 1996.

[FB91] GA Fichant and C Burks. Identifying potential trna genes in genomic DNA sequences. *J Mol Biol*, 220(3):659–71, 1991.

[FFHS00] C Flamm, W Fontana, IL Hofacker, and P Schuster. RNA folding at elementary step resolution. *RNA*, 6(3):325–38, 2000.

[FGLK+02] ST Fitz-Gibbon, H Ladner, UJ Kim, KO Stetter, MI Simon, and JH Miller. Genome sequence of the hyperthermophilic crenarchaeon Pyrobaculum aerophilum. *Proc. Natl. Acad. Sci.*, 99(2):984–989, 2002.

[FXM+98] A Fire, S Xu, MK Montgomery, SA Kostas, SE Driver, and CC Mello. Potent and specific genetic interference by double-stranded RNA in caenorhabditis elegans. *Nature*, 391(6669):806–11, 1998.

[GAH+03] Y Grad, J Aach, GD Hayes, BJ Reinhart, GM Church, G Ruvkun, and J Kim. Computational and experimental identification of c. elegans microrna. *Mol Cell*, 11(5):1253–63, 2003.

[GBE96] WN Grundy, TL Bailey, and CP Elkan. ParaMEME: a parallel implementation and a web interface for a dna and protein motif discovery tool. *Comput Appl Biosci.*, 4(12):303–310, 1996.

[GBK97] P Ganot, ML Bortolin, and T Kiss. Site-specific pseudouridine formation in preribosomal RNA is guided by small nucleolar rnas. *Cell*, 89(5):799–809, 1997.

[GCEB00] C Gaspin, J Cavaille, G Erauso, and JP Bachellerie. Archaeal homologs of eukaryotic methylation guide small nucleolar rnas: lessons from the pyrococcus genomes. *J Mol Biol*, 297(4):895–906, 2000.

[GFB+02] S Galardi, A Fatica, A Bachi, A Scaloni, C Presutti, and I Bozzoni. Purified box c/d snornps are able to reproduce site-specific 2'-o-methylation of target RNA in vitro. *Mol Cell Biol*, 22(19):6663–8, 2002.

[GH94] FJ Grundy and TM Henkin. Conservation of a transcription antitermination mechanism in aminoacyl-trna synthetase and amino acid biosynthesis genes in gram-positive bacteria. *J Mol Biol.*, 235(2):798–804, 1994.

[Gil86] W Gilbert. Origin of life- the RNA world. *Nature*, 319:618, 1986.

[GK95] S Guo and KJ Kemphues. par-1, a gene required for establishing polarity in c. elegans embryos, encodes a putative ser/thr kinase that is asymmetrically distributed. *Cell*, 81(4):611–20, 1995.

[GKA02] L Gitlin, S Karelsky, and R Andino. Short interfering RNA confers intracellular antiviral immunity in human cells. *Nature*, 418(6896):430–4, 2002.

[GL97] N Galtier and R Lobry, J. Relationships between genomic g+c content, rna secondary structures, and optimal growth temperature in prokaryotes. *J. Mol. Evol.*, 44:632–636, 1997.

[GL01] D Gautheret and A Lambert. Direct RNA motif definition and identification from multiple sequence alignments using secondary structure profiles. *J Mol Biol*, 313(5):1003–11, 2001.

[GLCF+92] P Girard, J, H Lehtonen, M Caizergues-Ferrer, F Amalric, D Tollervey, and B Lapeyre. Gar1 is an essential small nucleolar rnp protein required for pre-rrna processing in yeas. *EMBO J.*, 11(2):673–682, 1992.

[GMC90] D Gautheret, F Major, and R Cedergren. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for trna. *Comput Appl Biosci*, 6(4):325–31, 1990.

[Gra97] R Gras. Recherche multiple d'association d'expressions régulières flexibles dans

les grandes séquences génétiques. In *Publication interne - Projet Repco*. INRIA, 1997.

[Gro98] W Grogan, D. Hyperthermophiles and the problem of DNA instability. *Mol. Microbiol.*, 28(6):1043–1049, 1998.

[Gui02] F Guillonneau. *Recherche et identification de protéines reconnaissant les structures en triple-hélice d'acides nucléiques*. PhD thesis, Thèse de doctorat du Muséum national d'Histoire naturelle, 2002.

[Gus97] D Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[GW94] C Gaspin and E Westhof. The determination of the secondary structures of RNA as a constraint satisfaction problem. In Frontiers in Artificial Intelligence and Applications. IOS Press, editors, *Advances in Molecular Bioinformatics*. S. Schulze-Kremer, 1994.

[GWa03] E Gerhart, H Wagner, and J and, Vogel. *Noncoding RNAs: Molecular Biology and Molecular Medicine*, chapter Noncoding RNAs Encoded by Bacterial Chromosomes, pages 243–259. Jan Barciszewski and Volker A. Erdmann, Landes Bioscience, 2003.

[HAM03] R Hershberg, S Altuvia, and H Margalit. A survey of small rna-encoding genes in escherichia coli. *Nucleic Acids Res*, 31(7):1813–20, 2003.

[Han02] GJ Hannon. RNA interference. *Nature*, 418(6894):244–51, 2002.

[HBBH00] SM Hammond, E Bernstein, D Beach, and GJ Hannon. An rna-directed nuclease mediates post-transcriptional gene silencing in Drosophila cells. *Nature*, 404(6775):293–6, 2000.

[HEVM00] R Hoyne, P, M Edwards, L, A Viari, and J Maher, L. Searching genomes for sequences with the potential to form intrastrand triple helices. *J Mol Biol.*, 302(4):797–809, 2000.

[HFF+98] IL Hofacker, M Fekete, C Flamm, and Rauscher S Huynen, MA, Stolorz PE, and Stadler PF. Automatic detection of conserved RNA structure elements in complete RNA virus genomes. *Nucleic Acids Res*, 26(16):3825–36, 1998.

[Hig96] PG Higgs. Overlaps between RNA secondary structures. *Physical review letters*, 76(4):704–707, 1996.

[HS89] G Higgins, D and M Sharp, P. Fast and sensitive multiple sequence alignments on a microcomputer. *CABIOS*, 5:151–153, 1989.

[HZWF85] LM Hsu, J Zagorski, Z Wang, and MJ Fournier. Escherichia coli 6s RNA gene is part of a dual-function transcription unit. *J Bacteriol*, 161(3):1162–70, 1985.

[JCH95] I Jonassen, F Collins, J, and D Higgins. Finding flexible patterns in unaligned protein sequences. *Protein Science*, 4(8):1587–1595, 1995.

[Jon97] I Jonassen. Efficient discovery of conserved patterns using a pattern graph. *CABIOS*, 13:509–522, 1997.

[JTS02] JM Jacque, K Triques, and M Stevenson. Modulation of hiv-1 replication by RNA interference. *Nature*, 418(6896):435–8, 2002.

[KA03] P Ko and S Aluru. Space efficient linear time construction of suffix arrays. pages 200–210, 2003.

[KCM98] S Karlin, AM Campbell, and J Mrazek. Comparative DNA analysis across diverse genomes. *Annu Rev Genet.*, 32:185–225, 1998.

[KD93] C Kaplan, J and M Delpech. *Biologie moléculaire et médecine*, chapter Le génome des eucaryotes: le stockage de l'information., pages 15,43. Flammarion Médecine-Sciences, 1993.

[KEER02] F Kolb, C Ehresmann, B Ehresmann, and P Romby. Les ARN régulateurs : structure et mode d'action dans le contrôle de l'expression des gènes. *Regard sur la Biochimie*, 4:36–45, 2002.

[KFB+01] RF Ketting, SE Fischer, E Bernstein, T Sijen, GJ Hannon, and RH Plasterk. Dicer functions in RNA interference and in synthesis of small RNA involved in developmental timing in C. elegans. *Genes Dev*, 15(20):2654–9, 2001.

[KHvLP99] RF Ketting, TH Haverkamp, HG van Luenen, and RH Plasterk. Mut-7 of C. elegans, required for transposon silencing and RNA interference, is a homolog of Werne syndrome helicase and RNaseD. *Cell*, 99(2), 133-141 1999.

[Kis02] T Kiss. Small nucleolar rnas: an abundant group of noncoding rnas with diverse cellular functions. *Cell*, 109(2):145–8, 2002.

[KK00] RL Kelley and MI Kuroda. Noncoding RNA genes in dosage compensation and imprinting. *Curr Opin Genet Dev.*, 10(5):555–561, 2000.

[KLHB+96] Z Kiss-Laszlo, Y Henry, P Bachellerie, J, M Caizergues-Ferrer, and T Kiss. Site-specific ribose methylation of preribosomal rna: a novel function for small nucleolar rnas. *Cell*, 85(7):1077–88, 1996.

[KME02] RJ Klein, Z Misulovin, and SR Eddy. Noncoding rna genes identified in at-rich hyperthermophiles. *Proc Natl Acad Sci U S A*, 99(11):7542–7, 2002.

[KMP77] DE Knuth, JH Morris, and VR Pratt. Fast pattern matching in strings. *IAM Journal on Computing*, 6(1):323–350, 1977.

[Kol01] F Kolb. *Rôle de deux ARN dans le contrôle de l'expression des gènes: régulations de la réplication du plasmide R1 par un ARN anti-sens et des gènes de virulence de Staphylococcus areus par l'ARN III*. PhD thesis, Université Louis Pasteur Strasbourg I, 2001).

[KS03] J Karkkainen and P Sanders. Simple linear work suffix array construction. In *Proc. 30th Internat. Colloq. Automata, Languages and Programming*, pages 943–955, 2003.

[KSLK89] JD Kittle, RW Simons, J Lee, and N Kleckner. Insertion sequence is10 anti-sense pairing initiates by an interaction between the 5' end of the target RNA and a loop in the anti-sense RNA. *J Mol Biol*, 210(3):561–72, 1989.

[KSPP03] K Kim, D, S Sim, J, H Park, and K Park. Linear-time construction of suffix arrays. In *Proc. Fourteenth Annual Symp. Combinatorial Pattern Matching*, pages 186–199, 2003.

[Kur99] S Kurtz. Reducing the space requirement of suffix trees. *Software–Practice and Experience*, 29(13)(13):1149–1171, 1999.

[LA01] RC Lee and V Ambros. An extensive class of small rnas in caenorhabditis elegans. *Science*, 294(5543):862–4, 2001.

[LC04] D Laslett and B Canback. ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences. *Nucleic Acids Res.*, 32(1), 2004.

[LDM94] F Lisacek, Y Diaz, and F Michel. Automatic identification of group i intron cores in genomic DNA sequences. *J Mol Biol*, 235(4):1206–17, 1994.

[LE97] TM Lowe and SR Eddy. trnascan-se: a program for improved detection of transfer rna genes in genomic sequence. *Nucleic Acids Res*, 25(5):955–64, 1997.

[LE99] TM Lowe and SR Eddy. A computational screen for methylation guide snornas in yeast. *Science*, 283(5405):1168–71, 1999.

[LFA93] RC Lee, RL Feinbaum, and V Ambros. The c. elegans heterochronic gene lin-4 encodes small rnas with antisense complementarity to lin-14. *Cell*, 75(5):843–54, 1993.

[LGF+00] A Lescure, D Gautheret, D Fagegaltier, P Carbon, and A Krol. From RNA structure to the identification of new genes: The example of selenoprotein. *Journal of Health Science*, 46(6):409–413, 2000.

[LGY+03] LP Lim, ME Glasner, S Yekta, CB Burge, and DP Bartel. Vertebrate microrna genes. *Science*, 299(5612):1540, 2003.

[Lho93] O Lhomme. Consistency techniques for numeric CSPs. In *the 13-th International Joint Conference on Artificial Intelligence*, pages 232–238, 1993.

[LLW+03] LP Lim, NC Lau, EG Weinstein, A Abdelhakim, S Yekta, MW Rhoades, CB Burge, and DP Bartel. The microrna s of caenorhabditis elegans. *Genes Dev*, 17(8):991–1008, 2003.

[LLWB01] NC Lau, LP Lim, EG Weinstein, and DP Bartel. An abundant class of tiny rnas with probable regulatory roles in caenorhabditis elegans. *Science*, 294(5543):858–62, 2001.

[LQRLT01] M Lagos-Quintana, R Rauhut, W Lendeckel, and T Tuschl. Identification of novel genes coding for small expressed rnas. *Science*, 294(5543):853–8, 2001.

[LW01] NB Leontis and E Westhof. Geometric nomenclature and classification of RNA base pairs. *RNA*, 7(4):499–512, 2001.

[M97] Farach M. Optimal suffix tree construction with large alphabets. In *Proc. 38th Annual IEEE Symp. Foundations of Computer Science*, pages 137–143, 1997.

[Mac77] A Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118,, 1977.

[MAFM01] M Meli, B Albert-Fournier, and MC Maurel. Recent findings in the modern RNA world. *Int Microbiol*, 4(1):5–11, 2001.

[Mar84] HM Martinez. An RNA folding rule. *Nucleic Acids Res*, 12(1 Pt 1):323–34, 1984.

[Mar86] C Marvel, C. A program for the identification of tRNA-like structures in DNA sequence data. *Nucleic Acids Res.*, 10(1), 1986.

[Mau03] M-C Maurel. *Nouveaux débats sur le vivant*, chapter Origines de la vie, originalité du vivant, pages 9–21. Editions Kime, 2003.

[MBE+00] P Mourrain, C Beclin, T Elmayan, F Feuerbach, C Godon, JB Morel, D Jouette, AM Lacombe, S Nikic, N Picault, K Remoue, M Sanial, TA Vo, and H Vaucheret. Arabidopsis SGS2 and SGS3 genes are required for posttranscriptional gene silencing and natural virus resistance. *Cell*, 101(5):533–542, 2000.

[McC76] EM McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2):262–272, 1976.

[MEG+01] TJ Macke, DJ Ecker, RR Gutell, D Gautheret, DA Case, and R Sampath. Rnamotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Res*, 29(22):4724–35, 2001.

[MF90] S Mittal and B Falkenheimer. Dynamic Constraint Satisfaction Problem. In *AAAI'90*, 1990.

[MF95] ES Maxwell and MJ Fournier. The small nucleolar rnas. *Annu Rev Biochem*, 64:897–934, 1995.

[MGEW93] G Muller, C Gaspin, A Etienne, and E Westhof. Automatic display of RNA secondary structures. *Cabios*, 9(275):551–561, 1993.

[MH86] R Mohr and TC Henderson. Arc and path consistence revisited. *Source Artificial Intelligence*, 28(2):225–233, 1986.

[MM93a] U Manber and G Myers. Suffix arrays: A new method for on-line string searches. In *SIAM J. Comput.*, volume 22, 1993.

[MM93b] G Mehldau and G Myers. A system for pattern matching applications on biosequences. *Comput Appl Biosci*, 9(3):299–314, 1993.

[MMK01] M Matzke, J Matzke, A, and M Kooter, J. Rna: guiding gene silencing. *Science*, 293(5532), 2001.

[MP70] JH Morris and VR Pratt. A linear pattern-matching algorithm. Technical report, University of California, Berkeley, 1970. Technical Report 40.

[MS01] L Marsan and M-F. Sagot. Algorithms for extracting structured motifs using a suffix-tree with application to promoter and regulatory site consensus identification. *J. of Comput. Biol.*, 7:345–360, 2001.

[MTG+91] F Major, M Turcotte, D Gautheret, G Lapalme, E Fillion, and R Cedergren. The combination of symbolic and numerical computation for three-dimensional modeling of RNA. *Science*, 253:1255–1260, 1991.

[MTvGA95] E Morfeldt, D Taylor, A von Gabain, and S Arvidson. Activation of alpha-toxin translation in staphylococcus aureus by the trans-encoded antisense RNA, RNAIII. *EMBO J.*, 14(18):4569–4577, 1995.

[NMD+02] CD Novina, MF Murray, DM Dykxhoorn, PJ Beresford, RG Collman, J Lieberman, P Shankar, and PA Sharp. sirna-directed inhibition of hiv-1 infection. *Nat Med*, 8(7):681–6, 2002.

[Nor89] G North. Nobel prizes: chemistry. rna's catalytic role. *Nature*, 341(6243):556, 1989.

[NR02] G Navarro and M Raffinot. *Flexible pattern matching in strings*. University Press, Cambridge, 2002.

[NW70] SB Needleman and CD Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48(3):443–453, 1970.

[OLR+00] AD Omer, TM Lowe, AG Russell, H Ebhardt, SR Eddy, and PP Dennis. Homologs of small nucleolar rnas in archaea. *Science*, 288(5465):517–22, 2000.

[PCB+94] A Pavesi, F Conterio, A Bolchi, G Dieci, and S Ottonello. Identification of new eukaryotic trna genes in genomic DNA databases by a multistep weight matrix analysis of transcriptional control regions. *Nucleic Acids Res*, 22(7):1247–56, 1994.

[PDD+02] P Provost, D Dishart, J Doucet, D Frendewey, B Samuelsson, and O Radmark. Ribonuclease activity and RNA binding of recombinant human dicer. *EMBO J*, 21(21):5864–74, 2002.

[PF99] H Philippe and P Forterre. The rooting of the universal tree of life is not reliable. *J.Mol.Evol.*, 49:496–508, 1999.

[PH03] PJ Paddison and GJ Hannon. sirnas and shrnas: skeleton keys to the human genome. *Curr Opin Mol Ther*, 5(3):217–24, 2003.

[PL88] WR Pearson and DJ Lipman. Improved tools for biological sequence comparison. *Proc Natl Acad Sci USA.*, 85(8):2444–2448, 1988.

[PRS+00] AE Pasquinelli, BJ Reinhart, F Slack, MQ Martindale, MI Kuroda, B Maller, DC Hayward, EE Ball, B Degnan, P Muller, J Spring, A Srinivasan, M Fishman, J Finnerty, J Corbo, M Levine, P Leahy, E Davidson, and G Ruvkun. Conservation of the sequence and temporal expression of let-7 heterochronic regulatory rna. *Nature*, 408(6808):86–9, 2000.

[PSP86] L Pirtle, I, D Shortridge, R, and M Pirtle, R. Nucleotide sequence and transcription of a human glycine tRNAGCC gene and nearby pseudogene. *Gene*, 43(1-2):155–167, 1986.

[QMZ+01] H Qu, L, Q Meng, H Zhou, Q Chen, Y, Q Liang-Hu, M Qing, Z Hui, and C Yue-Qin. Identification of 10 novel snorna gene clusters from arabidopsis thaliana. *Nucleic Acids Res.*, 29(7), 2001.

[QNM+94] LH Qu, M Nicoloso, B Michot, MC Azum, M Caizergues-Ferrer, MH Renalier, and JP Bachellerie. U21, a novel small nucleolar RNA with a 13 nt. complementarity to 28s rrna, is encoded in an intron of ribosomal protein l5 gene in chicken and mammals. *Nucleic Acids Res*, 22(20):4073–81, 1994.

[Que94] Y Quentin. Emergence of master sequences in families of retroposons derived from 7sl rna. *Genetica*, 93(1-3):203–215, 1994.

[RE00a] E Rivas and SR Eddy. The language of rna: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–40, 2000.

[RE00b] E Rivas and SR Eddy. Secondary structure alone is generally not statistically significant for the detection of noncoding rnas. *Bioinformatics*, 16(7):583–605, 2000.

[RE01] E Rivas and SR Eddy. Noncoding rna gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(1):8, 2001.

[Ree00] R Reed. Mechanisms of fidelity in pre-mrna splicing. *Curr Opin Cell Biol.*, 12(3):340–345, 2000.

[RKJE01] E Rivas, RJ Klein, TA Jones, and SR Eddy. Computational identification of noncoding rnas in e coli by comparative genomics. *Curr Biol*, 11(17):1369–73, 2001.

[RM92] N Romano and G Macino. Quelling: transient inactivation of gene expression in

neurospora crassa by transformation with homologous sequences. *Mol. Microbiol.*, 6:3343–3353, 1992.

[RRL+02] MW Rhoades, BJ Reinhart, LP Lim, CB Burge, B Bartel, and DP Bartel. Prediction of plant microrna targets. *Cell*, 110(4):513–20, 2002.

[RSB+00] BJ Reinhart, FJ Slack, M Basson, AE Pasquinelli, JC Bettinger, AE Rougvie, HR Horvitz, and G Ruvkun. The 21-nucleotide let-7 RNA regulates developmental timing in caenorhabditis elegans. *Nature*, 403(6772):901–6, 2000.

[RWR+02] BJ Reinhart, EG Weinstein, MW Rhoades, B Bartel, and DP Bartel. Micrornas in plants. *Genes Dev*, 16(13):1616–26, 2002.

[SB92] M Singer and P Berg. *Gènes et génomes*, chapter Les molécules génétiques. Vigot, 1992.

[SBH+94] Y Sakakibara, M Brown, R Hughey, IS Mian, K Sjölander, RC Underwood, and D Haussler. Recent methods for RNA modeling using stochastic context-free grammars. In *Proceedings of the Asilomar Conference on Combinatorial Pattern Matching*, New York, NY, 1994. Springer-Verlag.

[Sch93] T Schiex. *Intelligence Artificielle et Informatique Théorique*, chapter Problèmes de satisfaction de contraintes, pages 227–266. CEPADUES, 1993.

[Sch02a] P Schattner. *Non-coding RNA*, chapter Computational gene-finding for non-coding RNAs. J Barcisewski (Landes Bioscience), 2002.

[Sch02b] P Schattner. Searching for RNA genes using base-composition statistics. *Nucleic Acids Res*, 30(9):2076–82, 2002.

[SD93] DB Searls and S Dong. A syntactic pattern recognition system for DNAsequences. In *Proceedings 2nd International Conference on Bioinformatics, Supercomputing, and Compelx Genome Analysis*, pages 89–101, 1993.

[Sea97] DB Searls. Linguistic approaches to biological sequences. *Comput Appl Biosci*, 13(4):333–44, 1997.

[Sea99] DB Searls. Formal language theory and biological macromolecules. In *Series in Discrete Mathematics and Theoretical Computer Science*, volume 47, pages 117–140, 1999.

[Sea02] DB Searls. The language of genes. *Nature*, 420(6912):211–7, 2002.

[SF94] D Sabin and E Freude. Contradicting conventional wisdom in constraint satisfaction. In *ECAI'94*, pages 125–129, 1994.

[SF99] DA Samarsky and MJ Fournier. A comprehensive database for the small nucleolar rnas from saccharomyces cerevisiae. *Nucleic Acids Res*, 27(1):161–164, 1999.

[SFV95] T Schiex, H Fargier, , and G Verfaillie. Valued Constraint Satisfaction Problems: hard and easy problems. In *Proc. of the International joint Conference in AI, Montreal, Canada*, 1995.

[SFV97] T Schiex, H Fargier, and G Verfaillie. Problèmes de satisfaction de contraintes valués. *Revue d'Intelligence Artificielle.*, 11(3), 1997.

[SGG96] DD Sledjeski, A Gupta, and S Gottesman. The small RNA, DsrA, is essential for the low temperature expression of rpos during exponential gro wth in Escherichia coli. *EMBO J.*, 15(15):3993–4000, 1996.

[Sha01] PA Sharp. RNA interference–2001. *Genes Dev*, 15(5):485–90, 2001.

[SKPP03] S Sim, J, K Kim, D, H Park, and K Park. Linear-time search in suffix arrays. In *Proc. Fourteenth Australasian Workshop on Combinatorial Algorithms*, pages 139–146, 2003.

[SLW+03] E Song, SK Lee, J Wang, N Ince, N Ouyang, J Min, J Chen, P Shankar, and J Lieberman. RNA interference targeting fas protects mice from fulminant hepatitis. *Nat Med*, 9(3):347–51, 2003.

[SS97] CM Smith and JA Steitz. Sno storm in the nucleolus: new roles for myriad small rnps. *Cell*, 89(5):669–72, 1997.

[Sta80] R Staden. A conputer program to search for trna genes. *Nucleic Acids Res.*, 8(4):817–825, 1980.

[Sta89] R Staden. Methods for calculating the probabilities of finding patterns in sequences. *Comput Appl Biosci*, 5(2):89–96, 1989.

[Sto90] GD Stormo. Consensus patterns in DNA. *Methods Enzymol.*, 183:211–21, 1990.

[Sto02] G Storz. An expanding universe of noncoding rnas. *Science*, 296(5571):1259, 2002.

[STS+00] L Simpson, OH Thiemann, NJ Savill, JD Alfonzo, and DA Maslov. Evolution of RNA editing in trypanosome mitochondria. *Proc Natl Acad Sci U S A.*, 97(13):6986–6993, 2000.

[SW81] TF Smith and MS Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1), 1981.

[THG94] D Thompson, J, G Higgins, D, and J Gibson, T. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680., 1994.

[TK97] D Tollervey and T Kiss. Function and synthesis of small nucleolar rnas. *Curr Opin Cell Biol*, 9(3):337–42, 1997.

[TS82] W Traub and L Sussman, J. Adenine-guanine base pairing ribosomal rna. *Nucleic Acids Res.*, 10:2701–2708, 1982.

[TS88] DH Turner and N Sugimoto. RNA structure prediction. *Annu. Rev. Biophys. Biophys. Chem.*, 17:167–192, 1988.

[TUM71] Jr Tinoco, I, OC Uhlenbeck, and Levine MD. Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293):362–367, 1971.

[Ukk92] E Ukkonen. Constructing suffix-trees on-line in linear time. In *Algorithms, Software, Architecture: Information Processing 92*, pages 484–492, 1992.

[Ukk95] E Ukkonen. On-line construction of suffix-trees. *Algorithmica*, 14:249–260, 1995.

[VBT+03] J Vogel, V Bartels, TH Tang, G Churakov, JG Slagter-Jager, A Huttenhofer, and EG Wagner. Rnomics in escherichia coli detects new srna species and indicates parallel transcriptional output in bacteria. *Nucleic Acids Res*, 31(22):6435–43, 2003.

[Via04] S Vialette. On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312(2-3), 2004.

[WC53] JD Watson and HC Crick. A structure for deoxyribose nucleic acid. *Nature*, 171:737, 1953.

[WGGN83] CR Woese, R Gutell, R Gupta, and HF Noller. Detailed analysis of the higher-order structure of 16s-like ribosomal ribonucleic acids. *Microbiol Rev*, 47(4):621–69, 1983.

[WHR93] B Wightman, I Ha, and G Ruvkun. Posttranscriptional regulation of the heterochronic gene lin-14 by lin-4 mediates temporal pattern formation in c. elegans. *Cell*, 75(5):855–62, 1993.

[WM91a] S Wu and U Manber. AGREP - a fast approximate pattern-matching tool. In *Proceedings of the Winter 1992 USENIX Conference San Francisco, USA*, pages 153–162. Berkeley, USA, 1991.

[WM91b] S Wu and U Manber. Fast text searching with errors. Technical report, University of Arizona, 1991.

[WM92] S Wu and U Manber. Fast text searching with errors. In *Communication ACM*, volume 35, pages 83–91, 1992.

[WRR+01] KM Wassarman, F Repoila, C Rosenow, G Storz, and S Gottesman. Identification of novel small rnas using comparative genomics and microarrays. *Genes Dev*, 15(13):1637–51, 2001.

[WS00] KM Wassarman and G Storz. 6S RNA regulates E. coli RNA polymerase activity. *Cell*, 101(6):613–623, 2000.

[XVGH03] P Xu, SY Vernooy, M Guo, and BA Hay. The drosophila microrna mir-14 suppresses cell death and is required for normal fat metabolism. *Curr Biol*, 13(9):790–5, 2003.

[ZS84] M Zuker and D Sankoff. Rna secondary structures and their prediction. *Bulletin of mathematical biology*, 46(4):591–621, 1984.

[ZTSB00] PD Zamore, T Tuschl, PA Sharp, and DP Bartel. Rnai: double-stranded RNA directs the ATP-dependent cleavage of mrna at 21 to 23 nucleotide intervals. *Cell*, 101(1):25–33, 2000.

Formalisme CSP (Constraint Satisfaction Problem) et localisation de motifs structurés dans les séquences génomiques

Résumé

La recherche d'occurrences de gènes d'ARN dans les séquences génomiques est un problème dont l'importance est renouvelée par la découverte récente de très nombreux ARN fonctionnels, opérant souvent en interaction avec d'autres molécules.

Le formalisme des réseaux de contraintes est approprié à cette problématique aussi bien sur le plan de la modélisation que sur les développements algorithmiques qu'il permet de proposer.

Après une analyse et une comparaison des outils existants plongés dans le cadre des réseaux de contraintes, nous montrons comment l'utilisation conjointe des réseaux de contraintes, des techniques de résolution associées et des algorithmes et structures de données du "pattern matching" permet de modéliser et de rechercher efficacement des motifs structurés en interaction (faisant intervenir plusieurs textes génomiques simultanément).

Mots-clés : motifs structurés, ARN, réseaux de contraintes.

CSP (Constraint Satisfaction Problem) formalism and localisation of structured motifs in genomic sequences

Abstract

Searching RNA genes occurrences in genomic sequences is a task which importance has been renewed by the recent discovery of numerous fonctional RNA, often interacting with other ligands.

The constraints network formalism is well adapted to this task, for molecules modeling as well as for the algorithmic developments it allows. After an analysis and comparison of existing tools within the framework of constraints networks, we show how the join use of constraints networks solving techniques and "pattern macthing" algorithms and data structures allows to model and efficiently search interacting structured motifs (with numerous genomic sequences interferring simultaneously).

Keywords : structured motifs, RNA, constraints networks.