Modèles combinatoires des structures d'ARN avec ou sans pseudonoeuds, application à la comparaison de structures.

Cédric Saule

UNIVERSITÉ PARIS-SUD 11

N° attribué par la bibliothèque
10147

THÈSE

pour obtenir le grade de

DOCTEUR de Université Paris-Sud 11

Spécialité : **Informatique**

préparée au laboratoire **Laboratoire de Recherche Informatique**

présentée et soutenue publiquement
par

Cédric Saule

le 17 décembre 2011

Titre:

Modèles combinatoires des structures d'ARN avec ou sans pseudonœuds, application à la comparaison de structures.

Directeur de thèse: **Alain Denise, PR Université Paris-Sud 11**

Jury

M. Alain Denise, PR Université Paris-Sud XI,	Directeur de thèse
M. Julien Allali, MCF Université Bordeaux I,	Examinateur
M. Serge Dulucq, PR Université Bordeaux I,	Rapporteur
Mme. Christine Gaspin, DR INRA Toulouse,	Rapporteur
Mme. Christine Paulin, PR Université Paris-Sud 11,	Président du jury
M. Jérôme Waldispühll, Assistant professor McGill Montréal,	Examinateur

Table des matières

Remerciements

Une thèse est un exercice de style scientifique, mais c'est aussi l'occasion de rencontrer des gens formidables et de partager des moments humains exceptionnels. Je tiens à remercier tous les gens qui ont compté et je présente par avance mes excuses à ceux que je pourrais oublier.

Je tiens tout d'abord à remercier les rapporteurs du présent manuscrit, Christine Gaspin et Serge Dulucq, pour leur travail d'évaluation ainsi que pour leurs questions et corrections très pertinentes. Je remercie également les autres membres de mon jury, Jérôme Waldispühl qui a fait un bien long chemin pour assister à ma soutenance et avec qui je travaillerai peut être à Montréal, Christine Paulin avec qui j'ai également eu le plaisir d'enseigner et Julien Allali avec qui j'ai eu la joie de collaborer dans le cadre du projet BRASERO.

Je remercie infiniment mon directeur de thèse Alain Denise. Je pense qu'avoir Alain comme directeur est une très grande chance pour un doctorant. C'est un scientifique de valeur qui possède en plus de grandes qualités humaines ; son humilité et sa générosité sont exemplaires. Alain a su me guider, me stimuler et me soutenir dans les pires moments et ce, bien au-delà de ce que l'on peut attendre d'un directeur de thèse. Plus qu'un directeur ou un collaborateur, je le considère comme un ami.

Je remercie chaleureusement Christine Froidevaux pour son accueil au sein de l'équipe bioinfo du LRI, pour sa confiance et son soutien ainsi que pour savoir créer et maintenir un environnement propice à la recherche dans l'équipe et l'université Paris-Sud XI.

Je remercie également Mireille Régnier avec qui j'ai eu le grand plaisir de collaborer. Je la remercie aussi pour son soutien, ses conseils, sa gentillesse et pour sa convivialité de bon aloi dans les moments de détente.

J'adresse un grand merci aux autres membres l'équipe bioinfo du LRI, Patrick Amar et Jérôme Azé, nos discussions sur la littérature de science fiction et le cinéma me manquent déjà, Sarah Cohen-Boulakia pour sa bonne humeur et sa jovialité ainsi que Sabine Pérès. Je remercie aussi Bastien Rance, Frédéric Lemoine, les supers BIBS de la première génération, mais aussi Lucie Gentils-Grandin, Philippe Rinaudo, Claire Herrbach, Lou Feng et Thomas Bourquart (Y a-t-il un biologiste dans cette salle Thomas ?) pour les nombreux moments de convivialité. Un grand merci également à Thomas Moncion pour son amitié et son grand sens de l'écoute.

Je remercie également Jean-Marc Steyaert pour m'avoir enseigné les bases du calcul asymptotique, nos échanges et notre collaboration. Je remercie Julie Bernauer qui n'a pas (complètement ?) réussi à m'effrayer avec la faune et la flore canadiennes. I thank Balaji Raman for interesting and kind discussions and for our

collaboration and I thank Saad Sheik who show me how to prepare Pakistanian tea and for nice discussions !

Je remercie également mes collaborateurs pour les enseignements, Frédéric Vernier, Burkhart Wolf, Sandrine Gouraud et Frédéric Voisin. Je remercie tout particulièrement Sophie Laplante, dont je découvre maintenant la belle province, pour son encadrement en tant que tutrice et ses précieux conseils pédagogiques.

Je remercie aussi Daniel Gautheret pour nos échanges constructifs sur les structures d'ARN et ses invitations au séminaire bioinfo de l'IGM, Bernard Labedan et Olivier Lespinet, Jean-Pierre Rousset pour son encadrement lors de mon stage de licence et nos nombreux échanges aux cours des années de thèse ainsi qu'Olivier Namy (Finalement Olivier, je n'élèverai pas des Yaks au Népal, ce sera des caribous au Canada !). Je remercie très chaleureusement Claire Toffano-Nioche pour sa présence, ses conseils, ses patientes relectures et surtout pour son amitié. Antonin Marchais, Stéphane Descorps-Declère, Fouad Boumezbeur pour l'amitié et les échanges de vision bioinformatique ainsi que Célia Floquet et Hélène Chommy.

Je remercie infiniment François Major pour le postdoc qu'il me permet de faire dans son équipe, la chaleur de son accueil, sa générosité et la grande confiance qu'il m'accorde. Je remercie également toute l'équipe du Major's lab de l'IRIC pour leur accueil et la convivialité. Je pense que nous aurons de beaux résultats ensemble. Je me plais à Montréal et c'est en grande partie grâce à eux.

Je remercie les participants du projet BRASERO, en particulier Yves D'Aubenton-Carafa, Fabrice Leclerc, Pascal Ferraro pour les échanges fructueux au sein de ce projet et Hélène Touzet, qui m'a également chaleureusement invité pour un séminaire du LIFL, Aïda Ouangraouda, pour nos échanges et notre collaboration présente.

Je remercie l'ensemble de la communauté ALEA pour cette formidable école d'été aussi instructive que conviviale, je remercie en particulier feu Philippe Flajolet pour sa générosité scientifique et notre discussion sur les partitions de nombres entiers. Je remercie également Philippe Duchon et Jean-François Marckert pour leur sympathie et leur convivialité.

Je remercie les membres du groupe JeBiF pour nos échanges associatifs, en particulier Magali Michaut et Virginie Bernard. Je leur souhaite à tous de devenir un jour de vieux bioinformaticiens !

Je remercie Julien David, avec lui je ne savais pas où nous commencerions la soirée mais je savais qu'elle finirait chez Léa, Florent Le Gac mon éternel "coloc" d'ALEA et Hayat Cheballah pour sa joie de vivre.

Je n'oublie pas non plus mes amies Anne-Laure Gaillard et Isabelle Le Claire qui savent garder un secret de belette sans perdre leur nez et chanter la carioca mais aussi Gilles Vieira en particulier pour ce JOBIM lillois mémorable.

Je remercie aussi tous mes amis grâce à qui je parvenais parfois à échapper à mes préoccupations scientifiques. Je remercie en particulier mes amis de toujours Lilian Thévenet, Pierre Le Tutour, Idriss Aberkane, Simine Khaladzadeh, dont les relectures ont bien contribué à l'amélioration de la forme de mon manuscrit et Mélanie Blounzi. Leur soutien indéfectible m'a énormément aidé durant ces années, peut être un jour pourrai-je les remercier comme ils le méritent. Je remercie Laura Giambruno, la sicilienne ou la parisienne … je ne sais plus, pour tous les moments

partagés. A tous, je souhaite que la distance ne modifie en rien la profondeur de nos relations.

Je remercie mes parents qui par le sacrifice inconditionnel de leur vie m'ont permis de me construire et d'avancer, il n'y a sans doute pas d'amour plus grand sur cette terre que celui-ci. Je remercie ma grand-mère et mes frères également pour leur soutien de tous les jours. Je pense que cette thèse a été plus difficile pour eux que pour moi.

Pour finir, je vais remercier Michel Termier et Guillaume Laroche. Michel a été celui qui m'a amené à la bioinformatique alors que je commençais mes études à Orsay en 1998. Il m'a amené à la bioinformatique en allant me chercher là où je n'avais rien à faire. Les longues discussions que nous avons eues, nos nombreux échanges ont très largement contribué à élaborer ma pensée, scientifique et non scientifique, actuelle. Il a cette humanité que seule une grande connaissance de soi-même peut apporter. Il termine sa carrière au moment où je commence la mienne et je ne sais exprimer toute l'admiration et la gratitude que j'ai pour lui. Guillaume était mon ami aussi depuis 1998, nos vies étaient symétriques pour bien des raisons. Ces maigres remerciements ne peuvent rendre hommage à tout ce qu'il m'a apporté. La plupart des gens ne voyaient de lui que sa force, physique et morale, ainsi que sa joie de vivre rarement altérée. Peu savaient ce contre quoi il se battait car il le faisait dans le silence et la dignité. Il y a des combats qui ne peuvent être remportés et il le savait depuis toujours. Lorsqu'il n'y a pas de mots, il vaut mieux se taire. A ces deux hommes, je dédie ces travaux.

Résumé

Ces travaux de thèse proposent une modélisation des structures secondaires d'ARN avec ou sans pseudonœuds. Selon une approche combinatoire, nous concevons différents modèles de ces structures que nous étudions sous deux aspects.

D'une part, nous définissons des modèles de génération aléatoire qui nous permettent de définir une mesure permettant une meilleure reconnaissance des structures biologiques. D'autre part, gr à des codages appropriés et des bijections vers des langages représentés par des grammaires non-contextuelles, nous dénombrons les structures composant l'espace de prédiction des algorithmes exacts de prédiction de structures secondaires avec pseudonœuds.

La première partie concerne des modèles aléatoires de structures d'ARN sans pseudonœuds. Nous montrons que ces structures aléatoires constituent une source de bruit pertinente lorsqu'il s'agit de déterminer si les logiciels de comparaison de structures attribuent un meilleur score à des comparaisons entre structures issues de la même famille d'ARN qu'à des alignements entre structures réelles et aléatoires. Nous comparons ensuite la sensibilité et la spcificité de RNAdistance, un programme de comparaison de structures, selon l'usage du score "brut" ou bien de la Z-valeur de ce score. Nous calculons plusieurs Z-valeurs selon différents modèles de structures aléatoires. Nous montrons que la Z-valeur calculée à partir d'un modèle de Markov améliore la détection des ARN de grande taille tandis que la Z-valeur calculée à partir d'un modèle basé sur des grammaires pondérées améliore la détection des ARN de petite taille.

Nous nous intéressons ensuite, dans une deuxième partie, aux algorithmes de prédiction de structure secondaire avec pseudonœuds. Nous complètons tout d'abord la classification de Condon *et al.* en décrivant les structures par leur graphe de cohérence et nous caractérisons également la restriction planaire de la classe de Rivas et Eddy. Nous étudions ensuite le compromis entre complexité des algorithmes existant et la taille de leur espace de prédiction. Nous dénombrons les structures en les codant par des mots de langages algébriques. Nous en déduisons alors des formules asymptotiques de dénombrement. Nous mettons aussi en évidence une bijection entre la classe de Lyngsø et Pedersen et des cartes planaires ainsi qu'une bijection entre la classe des pseudonœuds indifférenciés, que nous avons introduit, et les arbres ternaires. Nous montrons alors que les différences de compléxité observées des algorithmes de prédiction ne sont pas toujours justifiées par la taille de l'espace de prédiction. à partir de ces grammaires, nous concevons des algorithmes efficaces de génération aléatoire, uniforme ou non uniforme contrlée, de structures d'ARN avec pseudonœuds.

Abstract

This thesis proposes a model of RNA secondary structures with or without pseudoknots. According to a combinatorial approach, we design different models of these structures which we study according to two aspects. In one hand, we define random generation models which allow us to define a measure allowing a better recognition of biological structures. On the other hand, greatings to appropriated encodings and bijections to languages represented by non-contextual grammars, we count the structures composing the space of exact secondary structure prediction algorithms with pseudoknots.

The first part deals with random models of RNA structures without pseudoknots. We show that these structures are a relevant source of random noise when determining whether the structures comparison softwares attribute a better comparison score between structures from the same family than alignments between real and random structures. We then compare the sensitivity and spcificity of RNAdistance, a structures comparison software, depending on the use of the "raw" score or on the Z-value. We compute several Z-values according to different models of random structures. We show that the Z-value computed from a Markov model improves the detection of large RNA while the Z-value computed from a model based on weighted grammars improves the detection of small RNA.

We then consider, in the other hand, pseudoknotted secondary structure prediction algorithms. First we complete the Condon it et al. classification by describing the structures by their consistancy graph and we also characterize the planar restriction of the class of Rivas and Eddy. Then, we investigate the tradeoff between the complexity of existing algorithms and the size of their prediction space. We count the structures by coding them with words of algebraic languages. Then, we deduce asymptotic formulas count. We also show a bijection between the class of Lyngs o and Pedersen and planar maps and a bijection between the class of undifferentiated pseudoknots we introduced and ternary trees. We show that the observed differences in complexity prediction algorithms are not always justified by the size of the space prediction. From these grammars, we design efficient algorithms for generating random RNA structure, uniform or controlled non-uniform, with pseudoknots.

Introduction

Les molécules d'ARN sont des composants majeurs dans le fonctionnement de la cellule. Ils interviennent dans tous les processus cellulaires bien au-delà de ce que l'on croyait il y a une quinzaine d'années. Les ARN interviennent en particulier dans la régulation de l'expression génétique. La découverte de l'extinction de gènes par Fire et Mello en 1998 chez *C. elegans* [41] est un exemple phare de régulation de l'activité génique par des ARN. Cette découverte valut à leurs auteurs le prix Nobel de médecine et de physiologie en 2006. Ce mécanisme, qui a ensuite été découvert chez la plupart des animaux et des plantes, ouvre de nombreuses opportunités d'applications en biotechnologie et en médecine [47]. Alors que près de 80% du génome est transcrit chez les mammifères, seul 2% du génome correspond à des séquences d'ARN messagers matures qui seront traduits en protéines (ces dernières étaient alors considérées comme les actrices presque uniques de l'activité cellulaire) [69]. Ces découvertes récentes ont ouvert la voie à de nouveaux champs d'investigation en biologie ("RNomique") et en bioinformatique. La recherche sur les ARN a suscité un large intérêt de la communauté scientifique comme le montre l'enrichissement de la base de données Rfam [49] de 700 nouvelles familles en 2008. Cependant, la fonction de la majorité de ces ARN demeure encore inconnue. La fonction des ARN est fortement liée à leur structure spatiale. Afin de déterminer leur fonction, on cherche donc à élucider leur structure et à les identifier à des familles connues par homologie à l'aide de programmes de comparaison. La structure spatiale des ARN est obtenue par le repliement de la molécule sur elle même. Plusieurs niveaux de repliement intermédiaires peuvent être distingués. Nous nous intéressons ici à la structure secondaire des ARN. Ce niveau de repliement est intéressant car il permet déjà de déterminer la famille d'appartenance d'un ARN par comparaison de structure.

Nous posons donc dans un premier temps, en première Partie, les notions connues relatives aux structures d'ARN. Nous posons certaines normes de description des structures afin de clarifier et unifier les notions et les concepts qui les expriment. Nous définissons aussi les concepts combinatoires qui nous permettront par la suite de décrire les structures. Nous serons également amenés à introduire les concepts de génération aléatoire sur les langages algébriques.

Ces notions de bases étant posées, dans la seconde Partie, nous nous intéressons aux algorithmes de comparaison de structures secondaires sans pseudonœuds. Afin

de déterminer la fonction des molécules ou de dresser des phylogénies, les biologistes réalisent des alignements de séquences. Ces alignements renvoient, entre autres, des scores et/ou des distances qui permettent de déterminer si des séquences sont proches ou non. Afin de déterminer si deux séquences sont homologues ou non, il nous faut évaluer la significativité du score de comparaison. Pour cela, on peut utiliser des grandeurs statistiques comme la Z-valeur ou la p-valeur. Ces grandeurs nous permettent d'évaluer la significativité d'un score en le comparant à un score obtenu par hasard, le hasard étant modélisé par une distribution bien choisie sur l'espace des séquences.

Dans le cas des alignements de séquences, on sait, dans certains cas, calculer théoriquement ces grandeurs. Lorsque l'on ne sait pas les calculer, on doit alors recourir à la simulation. Mais le cas des comparaisons de structures d'ARN n'a pas encore été traité, c'est pourquoi nous sommes amené à proposer ici un modèle de calcul empirique de la Z-valeur pour les alignements d'ARN. Ce calcul repose sur l'obtention de structures aléatoires d'ARN. Pour être pertinent, un modèle de structures aléatoires doit capter les paramètres importants des structures réelles tout en restant assez libre pour que l'espace dans lequel on tire les structures soit suffisamment riche.

Dans un premier temps nous sommes amenés à définir plusieurs modèles possibles de structures aléatoires basés sur des modèles de Markov ou des grammaires algébriques pondérées, et dans un second temps, nous appliquons ces modèles dans deux contextes différents.

Dans une expérience préliminaire, nous voulons comparer les performances des logiciels existants de comparaisons de structures. Plus précisément, il s'agit de vérifier s'ils ont la capacité d'attribuer un meilleur score à des comparaisons de structures d'ARN réelles issues de la même famille biologique qu'à des comparaisons entre structures d'ARN réelles et des structures aléatoires qui représentent une source de bruit. La question que nous nous posons est de savoir s'il existe des sources de bruit plus pertinentes que d'autres. Nous montrons que certaines sources de bruit mettent davantage en difficulté les logiciels de comparaison de structures que d'autres.

Dès lors, nous nous intéressons au calcul de la Z-valeur. Nous définissons cette grandeur en nous basant sur les travaux réalisés pour les alignements de séquences. Nous comparons la sensibilité et la spécificité d'un programme de comparaison selon que l'on utilise le score "brut" ou bien la Z-valeur de ce score.

Dans ce but, nous calculons plusieurs types de Z-valeurs qui diffèrent par le modèle de structures aléatoires utilisées. Nous montrons que l'un de nos modèles basé sur des grammaires algébriques pondérées permet d'améliorer la détection des ARN de petite taille, le modèle de Markov, quant à lui permettant d'améliorer la détection des ARN de grande taille.

L'obtention des structures d'ARN repose soit sur la détermination de la structure réelle par des méthodes biochimiques (cristallographie, enzymes de restriction ...) soit sur des algorithmes de prédiction de structures à partir de leur séquence. Les méthodes biochimiques présentent toutefois le désavantage d'être complexes,

longues et onéreuses. En revanche, la prédiction de structure par des algorithmes s'avère plus rapide et moins onéreuse. Nous étudions, dans la troisième Partie, les algorithmes de prédiction de structure secondaire avec pseudonœuds. Les utilisateurs font généralement le choix de MFOLD ou de logiciels comparables lorsqu'ils ont besoin de replier des séquences d'ARN, cependant ces logiciels sont incapables de prédire les structures présentant des pseudonœuds. C'est pourquoi de nombreuses autres méthodes ont été développées pour prédire les structures avec pseudonœuds. Le problème de prédiction de structures secondaires avec pseudonœuds étant NP-Complet en toute généralité, les algorithmes polynomiaux ne peuvent prédire qu'un sous ensemble des structures possibles. Condon *et al.* [25] classent les algorithmes en comparant les ensembles de structures pouvant être prédites théoriquement par chacun d'eux. Convenons d'appeler *classe de structures* d'un algorithme donné l'ensemble des structures qu'il peut prédire en théorie. Condon *et al* ont montré qu'il existe, de ce point de vue, une hiérarchie stricte entre les algorithmes exacts : en d'autres termes, il existe des relations d'inclusion stricte entre leurs classes de structures. Condon *et al.* ont défini précisément ces classes en donnant pour chacune un système de ré-écriture qui permet de l'engendrer. Dans un premier temps, nous donnons une autre vision de ces classes en considérant leurs *graphes de cohérence*, notion introduite dans [53].

Puis nous nous intéressons aux relations entre ces classes et à la complexité des algorithmes de prédiction de structure qui leur correspondent. La complexité de chacun des algorithmes exacts de prédiction est connue. Alors que l'on pourrait s'attendre à ce que la hiérarchie de Condon *et al.* et la hiérarchie des complexités en temps soient clairement corrélées, on constate que ce n'est pas le cas. Plus précisément, nous soulevons le problème de quantifier, pour chaque classe, le compromis qui a été trouvé entre sa cardinalité et la complexité en temps de l'algorithme qui lui correspond. Afin d'évaluer ce compromis, il nous faut dénombrer chacune des classes. Dans ce but nous faisons appel à des approches de combinatoire énumérative et bijective. Pour toutes les classes sauf une, nous donnons un codage des structures par des mots de langages algébriques. A partir d'une grammaire non ambiguë du langage, nous obtenons des formules asymptotiques d'énumération. Nous montrons également l'existence de bijections entre certaines des classes et des classes de cartes planaires ou d'arbres ternaires, aboutissant ainsi à des formules closes de dénombrement. Finalement, nous mettrons en évidence le fait que les différences de complexité observées entre les différents algorithmes ne se justifient pas toujours par une différence significative du nombre de structures prédictibles.

Par ailleurs, le codage des classes par des langages algébriques nous permet de concevoir des algorithmes efficaces de génération aléatoire, uniforme ou non uniforme contrôlée, de structures d'ARN avec pseudonœuds.

Première partie

Notions biologiques et définitions préliminaires

Chapitre 1

Structure des ARN

Il nous est apparu que des équipes différentes utilisaient des notions et des concepts différents sur les structures d'ARN en se référant en fait à des éléments identiques. On peut citer notamment les notions de structure secondaire, de tige et de pseudonœud dont la définition est variable selon les auteurs. Etant donné ces différences, il nous est apparu qu'il y avait nécessité d'une approche normative.

Dans ce chapitre, nous allons donc rappeler et introduire les notions biologiques préliminaires et les définitions des structures des ARN (section 1.1, les modèles de repliement de ces molécules (section 2) et les familles d'ARN que nous utiliserons par la suite (section 3).

1 Structure secondaire

1.1 Définitions

L'ARN est un polymère linéaire constitué d'une séquence de nucléotides. Un nucléotide est un enchaînement d'un groupement phosphate, d'un sucre (le ribose) et d'une base azotée. Il existe quatre bases azotées dans l'ARN : l'adénine (A), la cytosine (C), la guanine (G) et l'uracyle (U). Ces bases peuvent former des liaisons hydrogène entre elles et provoquer ainsi le repliement de la molécule. Il existe plusieurs niveaux de structuration des ARN, que nous allons rappeler dans la liste de définitions suivantes :

Définition 1 (Structure primaire)**.** *La structure primaire d'un ARN est la séquence des nucléotides orientée de 5' vers 3'. Chaque résidu est identifié par la base qui le représente et par sa position* $1 \leq i \leq n$ *où n est le nombre total de résidus.*

Définition 2 (Appariement)**.** *Un appariement est une liaison créée entre deux résidus de la structure primaire. Par convention on nommera* (i, j) *l'appariement entre la base i et la base j. Par convention, on a* $i < j$*.*

Par abus de langage, on parlera aussi d'arc (i, j) pour parler de l'appariement (i, j) en référence à la représentation présentée dans le chapitre 1.2.

Définition 3 (Pied d'un appariement)**.** *Soit un appariement* (i, j)*. L'extrémité, ou pied, gauche est définie comme étant i, et son extrémité, ou pied, droite comme j.*

Définition 4 (Structure secondaire)**.** *La structure secondaire est le premier niveau de repliement des ARN. La structure secondaire consiste en l'appariement Watson Crick (liaison de A avec U et C avec G) ou Wobble (liaison de G avec U) des bases entre elles.*

Une structure secondaire est l'ensemble des appariements tels que, si (i,j) est un appariement alors $\nexists k$ tel que, (j,k) ou (k,j) ou (i,k) ou (k,i) soient un appariement.

Définition 5. *On peut définir trois sortes de relations entre les appariements :*

- *La relation d'empilement : (i,j) est empilé sur (k,l) si et seulement si $i<k<l<j$. On dit aussi que (k,l) est imbriqué sous (i,j). On note : $stack((i,j),(k,l)) = vrai$*
- *La relation de voisinage : (i,j) est voisin de (k,l) si et seulement si $i<j<k<l$ ou $k<l<i<j$. On note : $next((i,j),(k,l)) = vrai$.*
- *La relation de croisement : (i,j) et (k,l) se croisent si et seulement si $i<k<j<l$ ou $k<i<l<j$. On note : $cross((i,j),(k,l)) = vrai$*

Définition 6 (Structure secondaire sans pseudonœud)**.** *Dans une structure secondaire sans pseudonœud, $\forall (i,j),(k,l), cross((i,j),(k,l)) = faux$*

Définition 7 (Hélice ou tige.)**.** *Une hélice de taille k est une suite $(i,j),(i+1,j-1),\ldots(i+k-1,j-k+1)$ de bases appariées sans interruption.*

Définition 8 (Boucle)**.** *Une boucle est une région composée de bases non appariées. Ces boucles portent des noms différents selon leur emplacement dans la structure.*

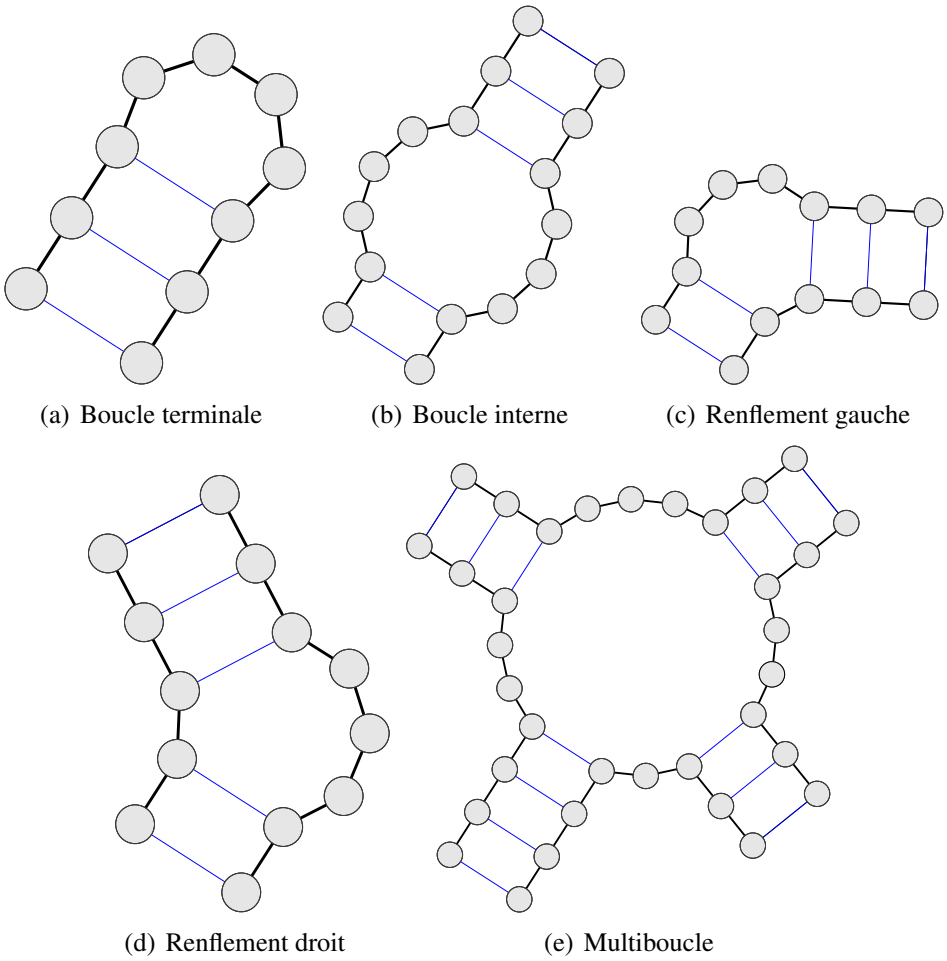
- *Boucle terminale : boucle fermant une hélice (Fig 1.1(a)).*
- *Boucle interne : les résidus non appariés interrompent les deux brins de l'hélice (Fig 1.1(b)).*
- *Renflement : résidus non appariés situés sur un seul brin d'une hélice. On parle de renflement gauche si les résidus se trouvent sur le brin coté 5' (Fig 1.1(c)) et de renflement droit s'ils se trouvent sur le brin coté 3' (Fig 1.1(d)).*
- *Multiboucle : résidus non appariés intercalant plusieurs démarrages de tiges (Fig 1.1(e)).*

Définition 9 (Graphe de cohérence [53])**.** *Le graphe de cohérence d'une structure secondaire d'ARN est le graphe qui associe à chaque appariement (i,j) un sommet, et tel que les sommets associés à (i,j) et (k,l) sont connectés si et seulement si $cross((i,j),(k,l)) = vrai$.*

Définition 10 (pseudonœud)**.** *Un pseudonœud est une composante connexe maximale non triviale du graphe de cohérence de la structure.*

Définition 11 (pseudonœud simple [106, 2])**.** *Un pseudonœud P est* simple *s'il existe deux nombres j_1 et j_2 avec $j_1 < j_2$, tels que :*

- *Chaque arc (i,j) de P satisfait soit $i < j_1 < j \leq j_2$ soit $j_1 \leq i < j_2 < j$.*
- *et si deux arcs (i,j) et (i',j') satisfont $i < i' < j_1$ ou $j_1 \leq i < i'$, alors $j > j'$.*



(a) Boucle terminale (b) Boucle interne (c) Renflement gauche

(d) Renflement droit (e) Multiboucle

FIGURE 1.1 – ***Sous-structures secondaires dans les ARN***

La première propriété assure que, pour chaque appariement de P, l'un de ses pieds exactement se situe entre j_1 et j_2. Les appariements sont divisés en deux ensembles : ceux qui ont leur autre extrémité plus petite que j_1, et ceux qui ont leur extrémité plus grande que j_2. Nous appelons ces deux ensembles respectivement, la *partie gauche* et la *partie droite* du pseudonœud. La seconde propriété de la définition assure que deux appariements du même ensemble ne peuvent se couper. La figure 1.2 montre un pseudonœud simple.

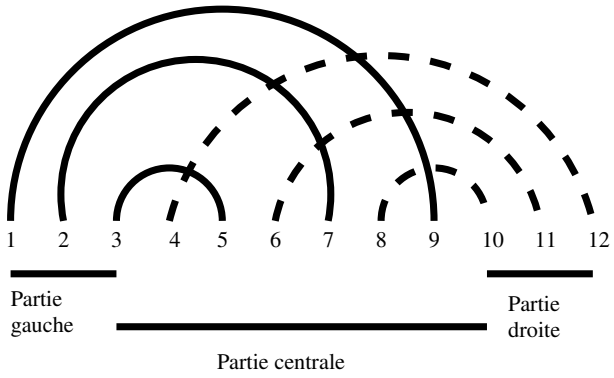


FIGURE 1.2 – ***Un pseudonœud donné par la séquence*** $(1,2,\ldots,12)$ ***et les appariements*** $(1,9)$, $(2,7)$, $(3,5)$, $(4,12)$, $(6,11)$, $(8,10)$. ***Ce pseudonœud est simple, avec*** $j_1 = 4$ ***et*** $j_2 = 9$.

Définition 12 (Structure pseudonouée)**.** *Une structure est dite pseudonouée si elle comporte au moins un pseudonœud. Une structure pseudonouée est dite simple si elle ne comporte que des pseudonœuds simples.*

Définition 13 (pseudonœud de type H [82])**.** *Un pseudonœud de type H est un pseudonœud simple dans lequel tous les appariements de la partie droite coupent tous les appariements de la partie gauche.*

Définition 14 (Epingles à cheveux embrassées)**.** *Deux épingles à cheveux embrassées forment un pseudonœud résultant d'appariements entre deux boucles terminales (Figure 1.3).*

1.2 Différentes représentations des structures d'ARN

On peut représenter de plusieurs façons les structures secondaires d'ARN. Nous allons voir que certaines représentations seront utilisées préférentiellement pour illustrer certaines propriétés. Nous rappelons ici les représentations suivantes :

- Projection planaire,
- Séquence arc-annotée,
- Graphe de corde,
- Mot de Motzkin,
- Liste d'appariements.

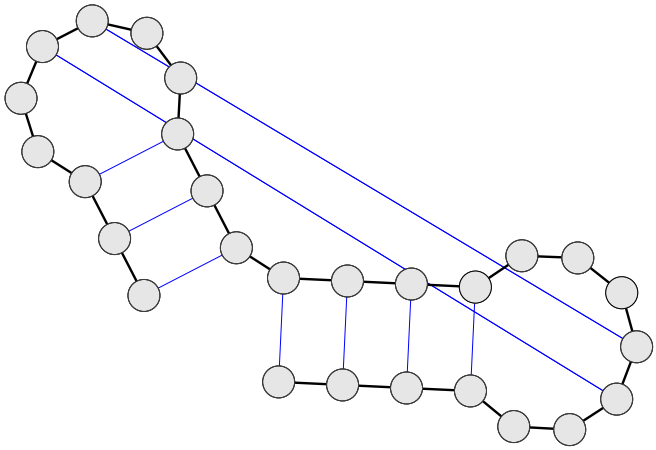


FIGURE 1.3 – ***Deux épingles à cheveux embrassées.***

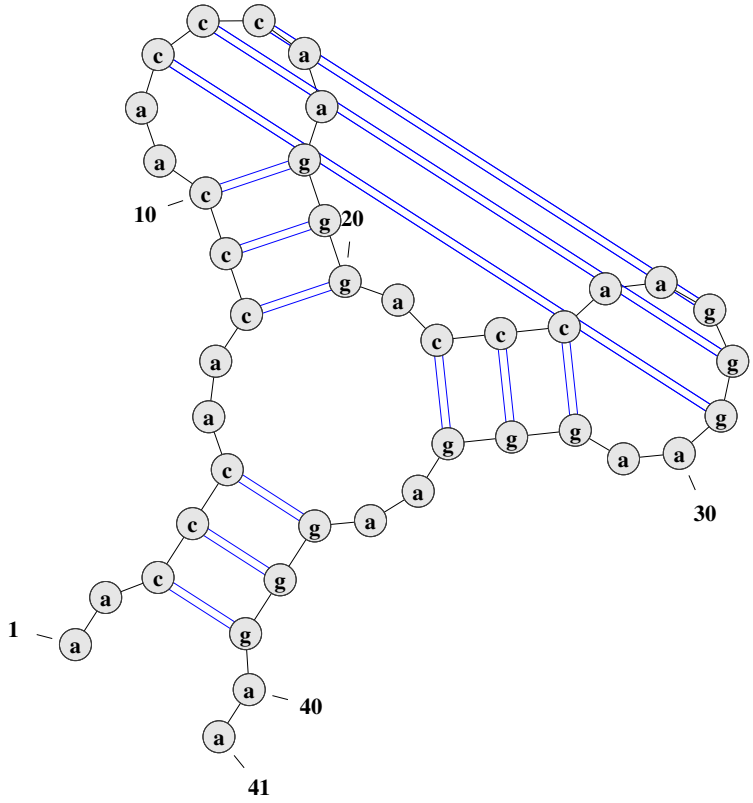


FIGURE 1.4 – ***Représentation en deux dimensions d'une structure d'ARN.***

Projection planaire

Cette représentation de l'ARN est une projection en deux dimensions de la structure secondaire (figure 1.4). Elle permet d'identifier aisément la forme générale de la structure et des différentes sous-structures.

Séquence arc-annotées

Dans cette représentation, la structure primaire est représentée sur une ligne. Deux bases sont reliées par un arc si elles sont appariées (Figure 1.5) et les arcs sont représentés du même coté de la ligne. Les pseudonoeuds sont facilement identifiables par des arcs qui se croisent.

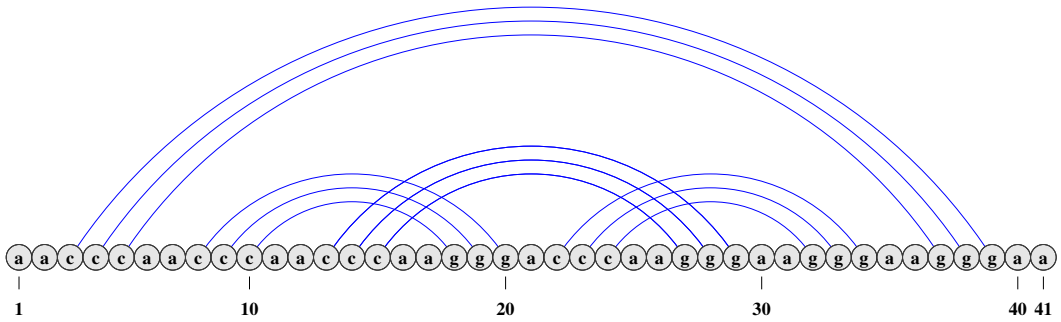


FIGURE 1.5 – ***Représentation arc-annotée d'une structure d'ARN.***

Graphe de corde

Cette représentation est obtenue à partir de la précédente en connectant la première base à la dernière (Figure 1.6). La structure primaire constitue la périphérie d'un cercle et la structure secondaire forme des cordes au sein de ce cercle. Les cordes qui se croisent définissent alors les pseudonoeuds.

Mot de Motzkin

Définition 15 (Mot de Motzkin)**.** *Le mot* $w \in \mathcal{A} = \{(,),.\}^*$ *est un mot de Motzkin si et seulement si pour tous les mots* $u \in \mathcal{A}^*$ *on a :*

- $|w|_(= |w|_)$
- *Pour tout mot u préfixe de w, on a* $|u|_(\geq |u|_)$

La combinatoire des mots de Motzkin est étudiée notamment dans [34, 102]. Les structures secondaires d'ARN sans pseudonœud peuvent être codées par des mots de Motzkin de la façon suivante :

- Si la base i est appariée à la base j avec $i < j$, on associe à la base i le symbole (et à la base j le symbole).
- Si la base i n'est pas appariée, on lui associe le caractère $\cdot$.
- Les pseudonœuds sont supprimés. Voir par exemple [98].

Le format Vienna est basé sur cette représentation :

```
>Nom_structure
aacccaacccaacccaagggacccaagggaagggaagggaa
..(((..(((.......))).(((.......))).))))..
```

On peut ajouter les bases qui connectent deux boucles (formant ainsi un pseudonoeud) en ajoutant de nouveaux types de parenthèses : $[,],\{,\},\ldots$. Les propriétés du langage ainsi formé seront exposées dans la section 1.1. Pour la structure présentée, on obtient ainsi :

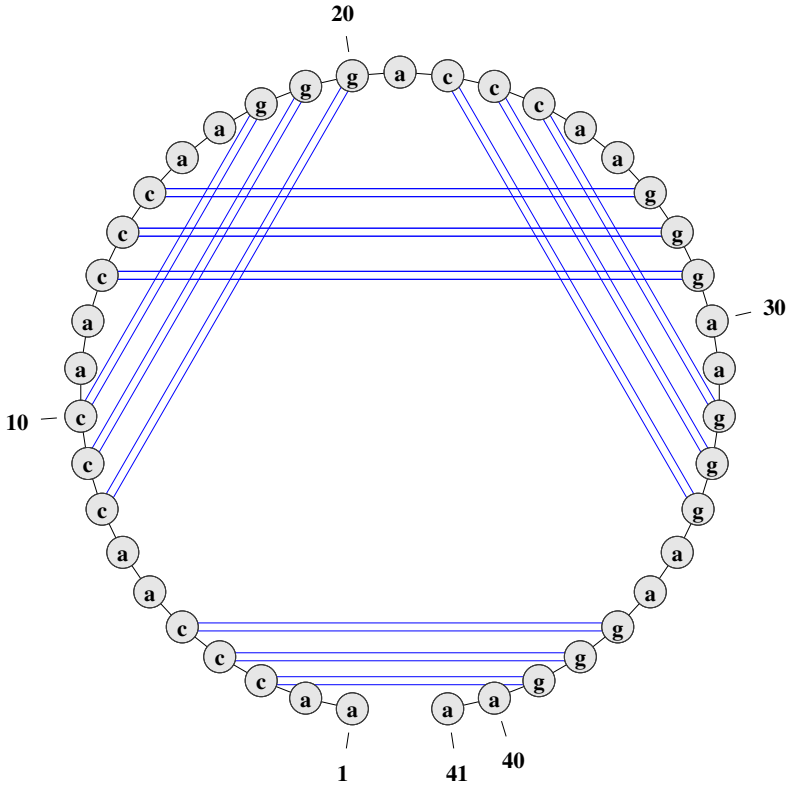


FIGURE 1.6 – ***Représentation sous forme de graphe de corde d'une structure d'ARN.***

```
>Nom_structure
aacccaacccaacccaagggacccaagggaagggaagggaa
..(((..(((..[[[..))).(((..]]]..))))..))))..
```

Liste d'appariements

Dans cette représentation, seules les bases appariées sont représentées sous forme de couples. Plusieurs formats de fichiers sont basés sur cette représentation comme bpseq, ct, etc... Exemple de fichier bpseq :

```
# Nom_structure
1 a 0
2 a 0
3 c 39
4 c 38
5 c 37
...
37 g 5
38 g 4
39 g 3
40 a 0
41 a 0
```

La séquence est indiquée au centre. A sa gauche, se trouve la position de la base et à sa droite, la position de la base à laquelle elle est appariée. Si la base n'est pas appariée, on note 0 dans la colonne de droite.

Arbres et graphes

On obtient la représentation sous forme d'arbre d'une structure secondaire sans pseudonoeud en associant à chaque base non appariée une feuille de l'arbre et à chaque paire de bases appariées un nœud interne (Figure 1.7). On enracine l'arbre sur un nœud supplémentaire que l'on nomme ε. La numérotation des nœuds se fait dans l'ordre du parcours en profondeur à main gauche de l'arbre.

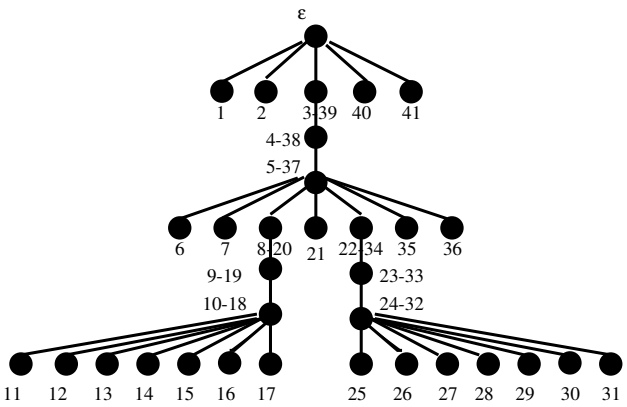


FIGURE 1.7 – ***Représentation sous forme d'arbre d'une structure d'ARN sans pseudonoeud.***

Lorsque l'on considère les pseudonoeuds, la représentation correspondante est un graphe. On l'obtient à partir de l'arbre correspondant à la structure secondaire en reliant les bases impliquées dans un pseudonoeud (Figure 1.8).

Cette représentation sous forme d'arbre nous permet une manipulation algorithmique plus aisée des structures secondaires. Par exemple, certains algorithmes d'alignement de structures sont basés sur des alignements d'arbres.

2 Modèle de repliement

La prédiction de structure *ab initio* consiste à déterminer à partir de la seule séquence d'un ARN, quel est l'ensemble des appariements dans la structure secondaire correspondante. Le principe de cette prédiction consiste à associer une fonction de coût à chaque structure puis à rechercher la structure qui optimise cette fonction de score. Elle peut correspondre à :

- Maximiser le nombre d'appariements dans la structure secondaire. Ce modèle est aussi appelé modèle de Nussinov [79].
- Minimiser l'énergie de repliement de la structure. Plusieurs modèles de calcul de l'énergie sont alors utilisés, pour les structures sans pseudonœud [74, 7] et pour les structures avec pseudonœuds [89, 33, 18].

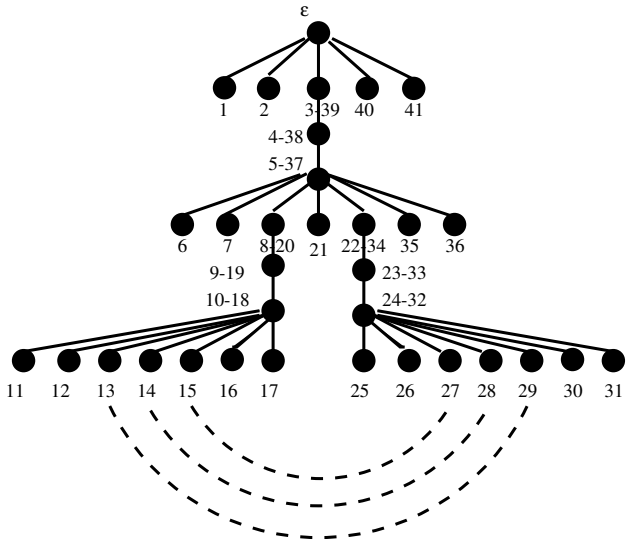


FIGURE 1.8 – ***Représentation sous forme de graphe d'une structure d'ARN avec pseudonoeud. Les arcs en pointillés représentent les liens entre les bases qui forment le pseudonoeud.***

– Maximiser la probabilité d'une structure connaissant l'ensemble de toutes les structures que peut adopter une séquence d'ARN. Le calcul de cette probabilité passe par le calcul de la *fonction de partition* selon un modèle d'énergie donné. Le calcul de fonction de partition à d'abord été réalisé pour les structures sans pseudonœud [75] puis avec pseudonœuds [33, 18].

Le repliement des ARN se déroule de façon hiérarchique, comme le montrent Tinoco et Bustamente dans [104]. Il repose sur le fait que le repliement se fait de façon à stabiliser les liaisons les plus fortes énergétiquement, puis les plus faibles. Ainsi, les appariements secondaires se forment en premier entre les bases les plus proches puis entre les bases les plus éloignées. Les pseudonœuds se formeraient alors pour stabiliser la structure secondaire. Les apparements non canoniques se forment en dernier car leur contribution énergétique est plus faible.

3 Différentes familles d'ARN structurés

Maintenant que nous avons défini les notions de structure des ARN avec ou sans pseudonœud, nous présentons les principales familles d'ARN structurés que nous utiliserons par la suite.

Micros ARN (miARN) Les micros ARN ont une taille d'environ 22 nucléotides. Ils interviennent dans la régulation post-transcriptionnelle. Ils sont structurés en tiges et en boucles. Ces ARN présentent une structure qui se résume essentiellement en une tige et une boucle (bien qu'il puisse y avoir des boucles interne et des renflements). La figure 1.9 donne un exemple de structures de micro ARN.

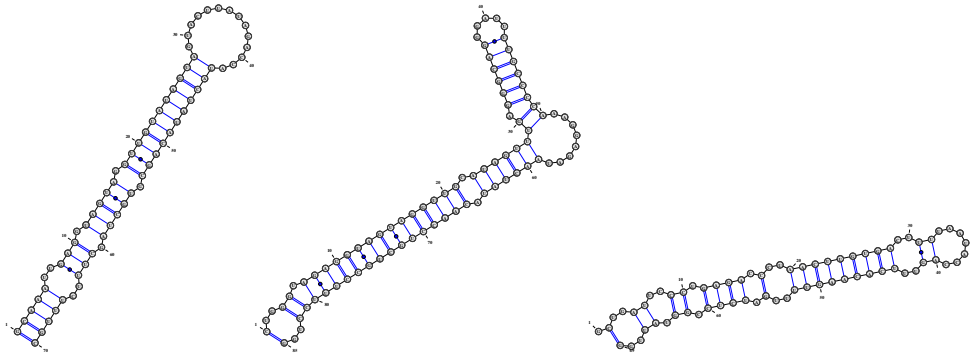


FIGURE 1.9 – ***Trois structures de micro ARN. Leur structure se résume essentiellement en une tige et une boucle.***

Petits ARN nucléolaires (ARNsno) Ces ARN interviennent dans la maturation des autres familles d'ARN. D'après [8], on distingue les petits ARN nucléolaires à boîte C/D qui interviennent dans la méthylation des bases, les petits ARN nucléolaires à boîte H/ACA qui interviennent dans la pseudouridylation.

Les ARN à boîte C/D ont une taille de 60-80 nucléotides [13] et sont en forme d'épingle à cheveux dont la tige ne fait qu'environ 5 résidus (figure 1.10). Les boîtes C (UGAUGA) et D (CUGA) sont positionnées respectivement à proximité des extrémités 5' et 3'. Une zone conservée complémentaire à l'ARN cible se situe à proximité de la boîte D.

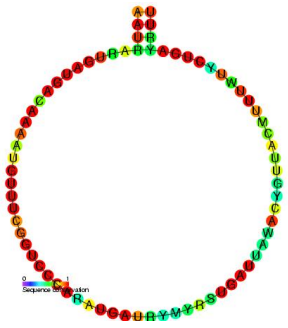


FIGURE 1.10 – ***Structure d'un ARN petit nucléaire à boîte C/D [49].***

Les ARN à boîte H/ACA font une taille d'environ 130 nucléotides [13]. Ils sont composés de deux épingles à cheveux connectées par une charnière et se terminant par un brin libre (figure 1.11). Ces ARN possèdent une boîte H (ANANNA) située sur la charnière et une boîte ACA (ACA) située sur le brin libre terminal. La reconnaissance de l'ARN cible se produit par complémentarité de bases avec les renflements des épingles à cheveux.

On peut notamment citer parmis les ARN à boîte H/ACA, les hgcG (High GC) chez les hyperthermophiles, dont le génome est riche en AT [65] (voir figure 1.12).

ARN de transfert (ARNt) Ces ARN sont chargés de transporter les acides aminés sur leur lieu d'assemblage dans le ribosome lors de la synthèse des polypeptides.

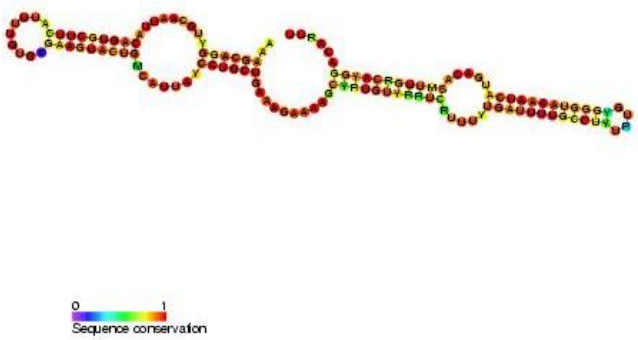


FIGURE 1.11 – ***Structure d'un ARN petit nucléaire à boîte H/ACA [49].***

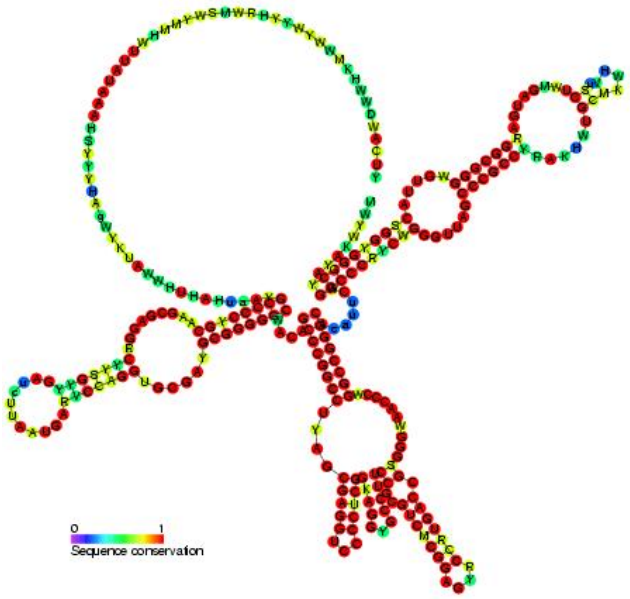


FIGURE 1.12 – ***Structure d'un hgCG (RF00064). Source Rfam [49]***

Les ARN de transfert ont une taille d'environ 70-80 nucléotides [13]. Ces ARN présentent de nombreuses modifications post-transcriptionnelles [73]. Leur structure secondaire en forme de trèfle est bien connue (figure 1.13) tandis que leur structure tridimensionnelle est en forme de L (Figure 1.14). Leur structure, contrairement à leur séquence est très conservée.

Ribonucléase P (RNase P) Ces ARN font partie d'une ribonucléoprotéine. Ils sont chargés du découpage des ARN en éléments plus petits. Nous présentons la structure secondaire de la RNAseP de *A. truei* dans la figure 1.15. Les RNase P présentent un pseudonœud qui connecte un renflement avec une multiboucle [28]. Cette structure n'est pas essentielle à l'activité catalytique du ribozyme mais contribue à son activité.

La ribonucléase P est un ribozyme chargée de la suppression de l'extrémité 5'

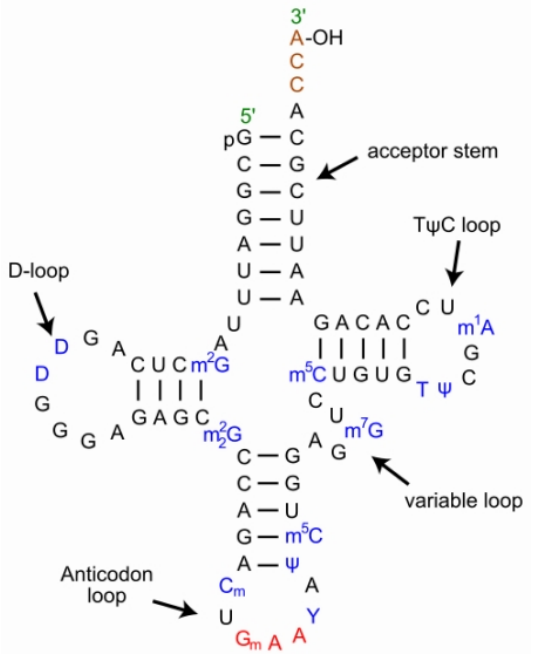


FIGURE 1.13 – ***Structure secondaire d'un ARN de transfert.***

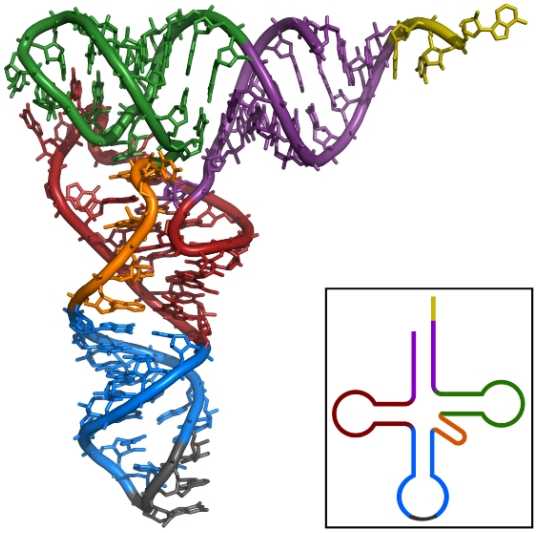


FIGURE 1.14 – ***Structure tridimensionnelle d'un ARN de transfert. Les correspondances avec la structure secondaire sont indiquées en bas à droite.***

du pré ARNt lors de sa maturation. Leur taille varie entre 220 et 440 nucléotides [13]. Leur structure varie selon les organismes.

Signal Recognition Particles (SRP) Ce complexe est composé de six protéines et d'un ARN, connu sous le nom de ARN 7SL. La taille de cet ARN varie de 300 à 400 nucléotides. La structure de cet ARN varie beaucoup selon les taxons [67]. Dans la figure 1.16, nous présentons la structure secondaire de l'ARN 7SL de O.

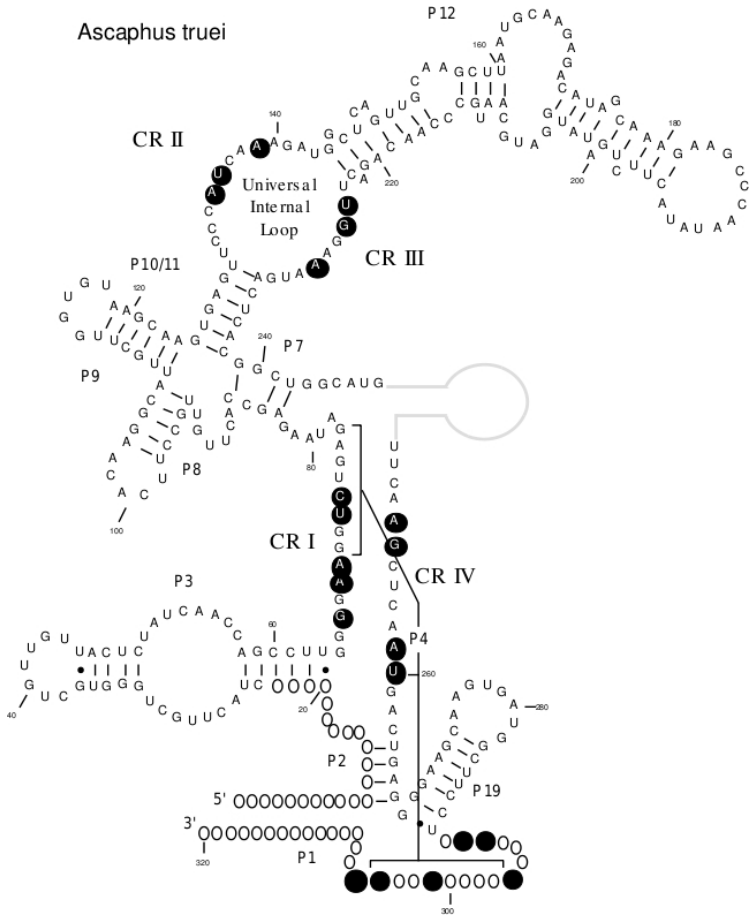


FIGURE 1.15 – ***Structure secondaire de la RNAseP de*** *A. truei* ***[14]***

sativa. Selon les taxons, les SRP ne possèdent pas toutes les hélices décrites dans la structure générale. Les SRP possèdent éventuellement des pseudonœuds selon les taxons, dont le rôle demeure encore incertain [28].

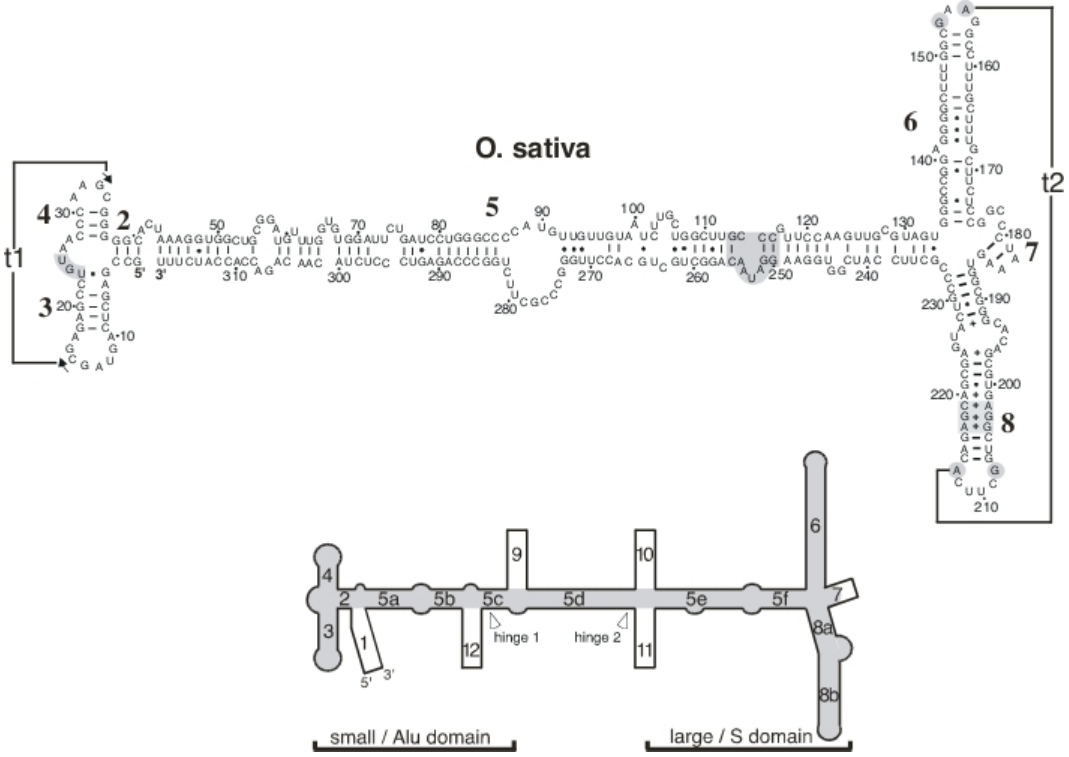


FIGURE 1.16 – ***En haut, structure secondaire de l'ARN 7SL de** O. sativa**. En bas, structure générale des SRP. Source : "SRP database" [6]***

Introns de groupe I Ces ARN sont des ribozymes capables d'auto-épissage sans apport d'énergie. Ils se trouvent dans les ARN messagers, les ARN de tranfert et les ARN ribosomaux. Contrairement à leur séquence, leur structure est bien conservée [94]. La structure de leur cœur catalytique se compose de neuf régions appariées (P1 à P9) [19, 112]. Le pseudonœud contribue à la stabilité de la structure cœur [28, 101]. Le repliement s'organise en deux domaines principaux, le domaine P4-P6 (P4, P5, P6 et P6a) et le domaine P3-P9 (P3, P7, P8 et P9) (voir la figure 1.17).

Introns de groupe II Ces introns sont des ribozymes capables de s'auto-épisser lentement à haute température et dans des conditions de salinité élevée [94]. Ces conditions font que les introns de groupe II ne peuvent s'épisser *in vivo* qu'avec l'aide d'une machinerie protéique. Tout comme pour les introns de groupe I, leur structure est plus conservée que leur séquence à l'exception des extrémités 5' (GUGYG) et 3' (AY). Leur structure se caractérise par six domaines (I à VI) en tige-boucle qui rayonnent autour d'un cœur central (voir figure 1.18). Des différences dans la séquence du domaine I permettent de distinguer deux sous-groupes d'introns de groupe II.

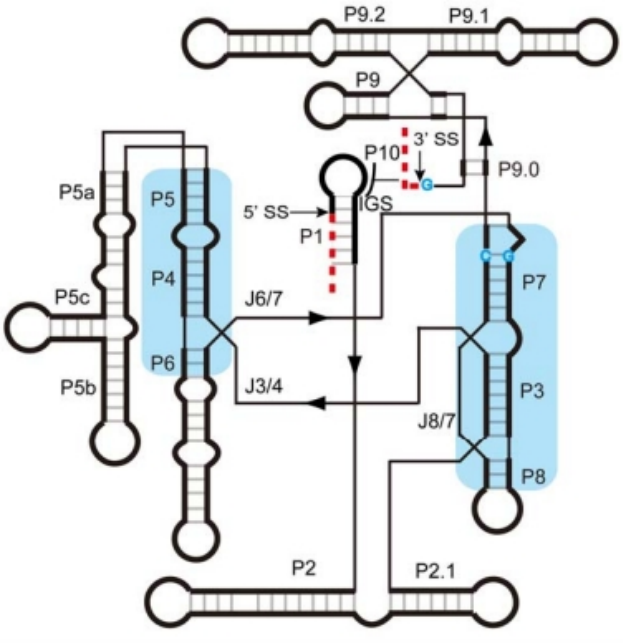


FIGURE 1.17 – ***Structure secondaire d'un intron de groupe I. Les exons sont marqués par les rayures rouges. SS marque le site d'épissage. Source : Haugen*** *et al* ***[54]***

ARN ribosomaux (rRNA) Ces ARN entrent dans la composition du ribosome. Le ribosome est constitué de deux sous-unités : la grande sous-unité et la petite sous-unité. Ces sous-unités sont composées de plusieurs ARN ribosomaux. Les ARNr 5S ont une taille de 115 nucléotides environ. Ils entrent dans la composition de la grande sous-unité du ribosome chez les eucaryotes et les procaryotes. Les ARNr 5,8S font une taille d'environ 150 nucléotides. Ils sont présents dans la grande sous-unité du ribosome chez les eucaryotes et interviennent dans la translocation du ribosome. Le 28S entre dans la composition de la grande sous-unité chez les eucaryotes. Cet ARN est l'homologue du 23S chez les procaryotes. Sa est d'environ 3000 nucléotides. Le 16S entre dans la composition de la petite sous-unité chez les prokaryotes, sa taille est d'environ 1500 nucléotides. Le 18S est l'homologue du 16S chez les eucaryotes.

Plusieurs pseudonœuds très conservés en termes de structure et de séquence sont présents dans la petite sous-unité du ribosome [28]. Ils interviennent dans la stabilité du ribosome lors de son assemblage ainsi que dans la précision de la traduction. Leur mutation entraine une sensibilité accrue à la streptomycine ce qui indique que ces pseudonœuds contribuent à la stabilité du ribosome. De nombreux pseudonœuds sont présents dans la grande sous-unité interviennant dans la stabilité des sites de reconnaissance.

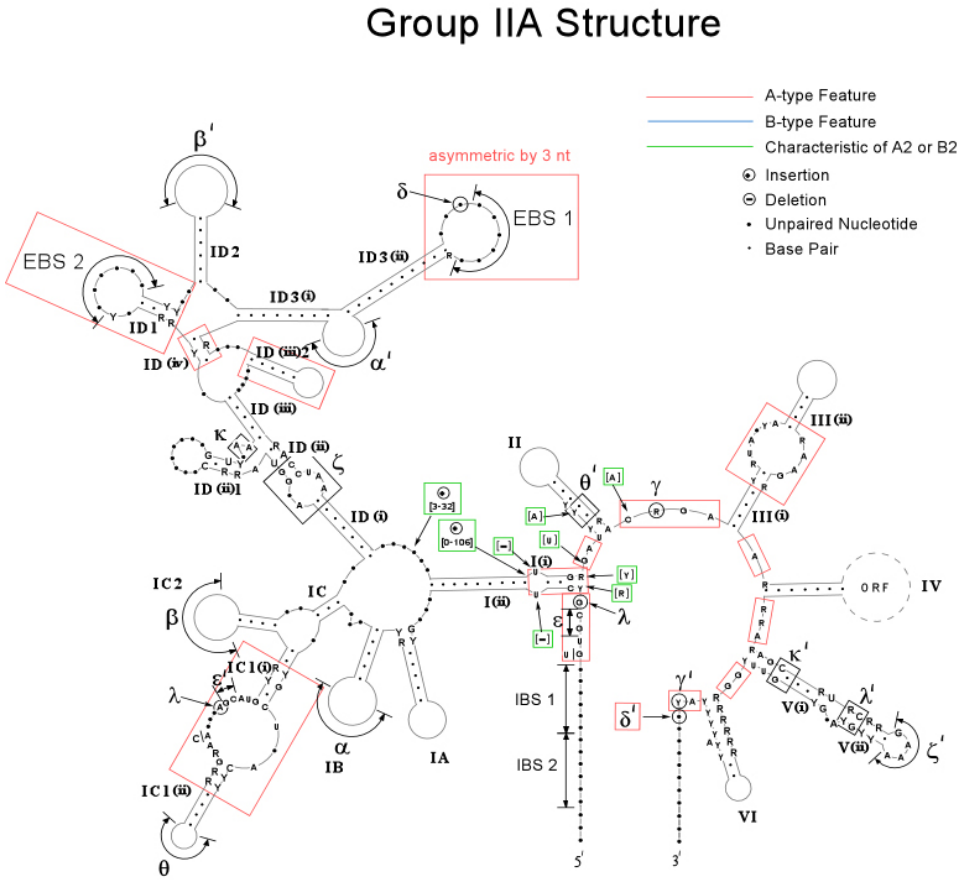


FIGURE 1.18 – ***Structure secondaire d'un intron de groupe IIA. Source : "Database for mobile group II introns" [27].***

Chapitre 2

Théorie des langages et combinatoire

Ainsi que nous l'avons annoncé dans l'introduction, nous allons nous intéresser par la suite au dénombrement et à la génération aléatoire de structures secondaires d'ARN. Les structures secondaires d'ARN sans pseudonœud peuvent être codées par des mots appartenant à des langages algébriques. Nous savons énumérer et engendrer aléatoirement, de façon uniforme ou non uniforme contrôlée les mots issus de ces langages. Dans cette perspective, nous développons ci-dessous les rappels nécessaires à la compréhension de ces notions en matière de théorie des langages, de génération aléatoire et de calcul asymptotique.

1 Langages et classes combinatoires

Nous croyons nécessaire d'introduire ici, quelques notions de théorie des langages, afin de présenter les grammaires algébriques. Nous introduirons ensuite les notions de dénombrement puis de génération aléatoire qui sont des concepts qui sous-tendent les travaux qui suivent.

1.1 Notions de théorie des langages

Définition 16. *Soit un alphabet* Σ*. Un langage* $\mathcal{L}$ *est un ensemble, possiblement infini, de mots sur* Σ^**.*

Nous nous intéressons ici aux langages algébriques. Ces langages sont reconnus par des automates à pile ou engendrés par des grammaires non contextuelles. Une grammaire est un mécanisme de ré-écriture à base de règles qui engendrent, par dérivations successives à partir d'un symbole initial, un langage.

Définition 17 (Grammaire non contextuelle)**.** *Soit la grammaire* $\mathcal{G}$ *définie par le quadruplet suivant :* $(\Sigma, \mathcal{N}, S, \delta)$ *où* Σ *est l'alphabet terminal,* $\mathcal{N}$ *l'alphabet non-terminal, S le symbole non-terminal initial appelé axiome et* δ *l'ensemble des règles de ré-écriture (on les appelle aussi règles de production). Une grammaire est dite algébrique ou non contextuelle si* δ *est défini de la façon suivante :* $\delta \subset \mathcal{N} \times (\mathcal{N} \cup \Sigma)^*$*. Les règles de production sont donc de la forme :* $P \to \alpha$ *avec* $\alpha \in (\mathcal{N} \cup \Sigma)^*$ *et* $P \in \mathcal{N}$*.*

Voici par exemple la grammaire non contextuelle engendrant le langage de Motzkin : $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta)$, $\Sigma = \{., (,)\}$, $\mathcal{N} = \{S\}$.

Les règles de production sont :

$$\delta = \begin{cases} S & \rightarrow & (& S &) & S \\ S & \rightarrow & .S & & & \\ S & \rightarrow & \varepsilon & & & \end{cases}$$

Définition 18 (Dérivation directe)**.** *Soit $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta)$. On dit qu'un mot $\beta \in (\Sigma \cup \mathcal{N})^*$ dérive directement d'un mot $\alpha \in (\Sigma \cup \mathcal{N})^*$ si et seulement si $\alpha = a_p P a_s$ et $\beta = a_p \omega a_s$ avec $(P \rightarrow \omega) \in \delta$. On note alors $\alpha \xrightarrow{\mathcal{G}} \beta$.*

Prenons par exemple le langage des mots de Motzkin et considérons la suite de dérivations suivante :

$S \rightarrow (S) S \rightarrow ((S)) S \rightarrow (( )) S \rightarrow (( ))$

Voici la liste des dérivations directes de notre exemple :

- $S \rightarrow (S) S$
- $(S) S \rightarrow ((S)) S$
- $((S)) S \rightarrow (( )) S$
- $(( )) S \rightarrow (( ))$

Définition 19 (Dérivation)**.** *Dans $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta)$, un mot $\omega \in \Sigma^*$ dérive d'un non terminal $P \in \mathcal{N}$ si et seulement s'il existe $s_1, \ldots, s_k \in (\mathcal{N} \cup \Sigma)^*, k \in \mathbb{N}$ tels que : $P \xrightarrow{\mathcal{G}} s_1 \xrightarrow{\mathcal{G}} \ldots s_k \xrightarrow{\mathcal{G}} \omega$. On note $P \xrightarrow[\mathcal{G}]{*} \omega$*

Définition 20 (Langage associé à une grammaire)**.** *On appelle langage d'un non terminal P dans $\mathcal{G}$, qu'on note $\mathcal{L}_{\mathcal{G}}(S)$, l'ensemble des mots terminaux dérivables à partir du non terminal P dans $\mathcal{G}$:*

$$\mathcal{L}_{\mathcal{G}}(P) = \{\omega \in \Sigma^* | P \xrightarrow[\mathcal{G}]{*} \omega\}$$

Le langage $\mathcal{L}(\mathcal{G})$ associé à une grammaire $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta)$ est le langage associé à l'axiome S dans $\mathcal{G}$:

$$\mathcal{L}(\mathcal{G}) = \mathcal{L}_{\mathcal{G}}(S)$$

Définition 21 (Grammaire non ambiguë)**.** *Une grammaire $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta)$ est dite non ambiguë si tout mot ω reconnu par cette grammaire est obtenu par une unique dérivation à partir de l'axiome S où le non terminal le plus à gauche est ré-écrit à chaque étape.*

Dans la suite du document et dans un souci de simplification des notations, on omettra de mettre en indice les dérivations et langages par la grammaire considérée quand celle-ci est implicite.

1.2 Classes combinatoires et dénombrement

Nous allons définir la notion de classe combinatoire, puis présenter les méthodes de dénombrement de ces objets.

Définition 22 (Fonction de taille)**.** *On appelle fonction de taille la fonction* $|.| : \Sigma^* \to \mathbb{N}$ *définie récursivement par :*

$$|\varepsilon| = 0.$$

$$|\alpha| = 1, \alpha \in \Sigma$$

$$|\omega.\omega'| = |\omega| + |\omega'|, \omega, \omega' \in \Sigma^*$$

Définition 23 (Classe combinatoire)**.** *Une classe combinatoire* $\mathcal{C}$ *est un ensemble muni d'une fonction de taille* $|.| : \mathcal{C} \to \mathbb{N}$*, qui induit des sous-ensembles finis* $\mathcal{C}_n$ *d'objets de* $\mathcal{C}$ *de taille n.*

Les langages constituent une classe combinatoire très naturelle dans laquelle l'opérateur de taille $|.|$ est associé à l'opérateur de concaténation.

Définition 24 (Série génératrice de dénombrement)**.** *Soit* $\mathcal{C}$ *une classe combinatoire et* $c_n = |\mathcal{C}_n|$ *le nombre d'objets de taille n dans* $\mathcal{C}$*, alors on appelle série génératrice de dénombrement de* $\mathcal{C}$ *la série formelle* $\mathcal{C}(z)$ *telle que :* $C(z) = \sum_{o \in \mathcal{C}} z^{|o|} = \sum_{n \geq 0} c_n z^n$

On peut obtenir une telle série grâce à une décomposition non ambiguë de la classe combinatoire. Donc à partir d'une grammaire non ambiguë, on peut construire un système d'équations dont la solution est la série génératrice. La méthode utilisée est baptisée DSV.

Définition 25 (Méthodologie DSV)**.** *Soit* $\mathcal{C}$ *une classe combinatoire, la méthodologie DSV est une méthode générale pour la détermination de la série génératrice associée à la classe* $\mathcal{C}$*. Elle consiste en l'application des 3 points ci-dessous :*

- ***Encodage de*** $\mathcal{C}$ ***:*** *Trouver un langage algébrique* $\mathcal{L}$ *décrit par une grammaire G non ambiguë telle que* $\mathcal{L}_n$ *soit en bijection avec* $\mathcal{C}_n$ *pour tout n.*

- ***Construction d'un système d'équations :*** *On transpose le système de règles de* $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta)$ *en équations sur les séries génératrices* $C_i(z)$ *associées aux non terminaux* C_i*.*

$$\begin{cases} C_1 & \to & \alpha_1^1 | \ldots | \alpha_1^{m_1} \\ C_2 & \to & \alpha_2^1 | \ldots | \alpha_2^{m_2} \\ & \ldots & \\ C_k & \to & \alpha_k^1 | \ldots | \alpha_k^{m_k} \end{cases} \Rightarrow \begin{cases} C_1(z) & = & \phi(\alpha_1^1) + \ldots + \phi(\alpha_1^{m_1}) \\ C_2(z) & = & \phi(\alpha_2^1) + \ldots + \phi(\alpha_2^{m_2}) \\ & \ldots & \\ C_k(z) & = & \phi(\alpha_k^1) + \ldots + \phi(\alpha_k^{m_k}) \end{cases}$$

où $k = |\mathcal{N}|, C_i \in \mathcal{N}, \alpha_j^i \in (\Sigma \cup \mathcal{N})^*, C_j(z) = \sum_{o \in \mathcal{L}(C_j)} z^{|o|}$ *avec* ϕ *:*

$$\phi(\varepsilon) = \varepsilon$$

$$\phi(t\omega) = z.\phi(\omega), t \in \Sigma$$

$$\phi(C_i\omega) = C_i(z).\phi(\omega), C_i \in \mathcal{N}$$

– ***Résolution du système d'équations***
L'élimination est toujours possible (système d'équations algébriques), on peut donc se ramener à une unique équation impliquant $C_i(z)$ *et* z *pour un i donné. Celle-ci peut cependant s'avérer d'un degré élevé en* C_i *, ce qui rend la détermination de* $C_i(z)$ *potentiellement problématique. Soit* $C_\alpha := S$ *l'axiome de la grammaire, on a alors :*

$$\mathcal{L}(S)(z) = \mathcal{L}(\mathcal{G})(z) = \mathcal{L}(z)$$

Calculons la série génératrice des mots de Motzkin à titre d'exemple :

$$S \rightarrow (\ S \ ) \ S \mid . \ S \mid \varepsilon \Rightarrow \quad S(z) = z*S(z)*z*S(z) + z*S(z) + 1$$

On résoud le système de façon à obtenir l'équation suivante : $z^2S^2(z) + (z-1)S(z) + 1 = 0$

On en déduit la série suivante : $S(z) = \frac{1-z-\sqrt{1-2z-3z^2}}{2z^2}$

D'après cette définition, on comprend que l'on compte en réalité le nombre de dérivations qui permettent d'engendrer un mot et non le nombre de mots à proprement parler. La description d'un langage par une grammaire non ambiguë est dès lors indispensable pour le dénombrement par cette méthode. A partir de la série génératrice, nous pouvons extraire les coefficients C_n ou en calculer un équivalent asymptotique pour $n \rightarrow \infty$

Nous disposons donc maintenant d'un outil pour dénombrer les langages. Nous allons maintenant étudier des modèles de génération aléatoires de langages algébriques basés sur leur grammaire associée.

2 Génération aléatoire de mots de langages algébriques

Afin de comprendre les fondements sur lesquels repose GenRGenS [84], le logiciel de génération aléatoire que nous utiliserons par la suite, nous allons présenter les grands principes de la génération aléatoire ainsi que les deux principaux modèles de génération de mots basés sur des grammaires algébriques. Un exposé de ces méthodes peut également être trouvé dans [83]

Méthode récursive

Cette méthode repose sur une définition récursive de l'objet que l'on veut engendrer. A chaque étape de l'algorithme, on effectue un choix local qui détermine la suite de la génération. Cette méthode se déroule en deux étapes : Une étape de précalcul dans laquelle on calcule les coefficients des différentes séries génératrices, et une phase de génération dans laquelle on engendre les objets de façon uniforme. Le coût de cette génération est exponentiel par rapport au nombre maximal de non terminaux apparaissant dans la partie droite des règles de production. Afin de réduire ce coût, on utilise une variante de la forme normale de Chomsky qui ne fait apparaître qu'au plus deux non terminaux dans la partie droite.

Définition 26 (Forme Normale de Chomsky (FNC))**.** *Une grammaire $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta)$ est en forme normale de Chomsky si et seulement si, pour tout $P \in \mathcal{N}$ on a $(P \to \alpha) \in \delta$ avec $\alpha = t$ ou $\alpha = P'P''$*

Avec $t \in \Sigma, P', P'' \in \mathcal{N}/S$. On peut avoir $S \to \varepsilon$ si la grammaire peut engendrer le mot vide.

Une forme normale de Chomsky peut être calculée pour toute grammaire non contextuelle.

Nous travaillons avec une variante de la forme normale de Chomsky dont les règles de production se présentent de la façon suivante :

- $P \to t$
- $P \to P'P''$
- $P \to P'|P''$ (Ceci n'est pas à proprement parler une règle mais la conjonction de deux règles.)

Avec $t \in \Sigma, P', P'' \in \mathcal{N}/S$. On peut avoir $S \to \varepsilon$ si la grammaire peut engendrer le mot vide.

On peut calculer à partir de cette forme le nombre de mots dans le langage de la façon suivante :

Théorème 1. *Dans une grammaire $\mathcal{G}$ non-ambiguë en FNC, les langages $\mathcal{L}(S)$ et les nombres p_k de mots de taille k issus de P obéissent aux relations suivantes :*

$$\begin{array}{lrc} \textit{règles d'union} & & P \to P'|P'' \\ & \Rightarrow & \mathcal{L}(P) = \mathcal{L}(P') \cup \mathcal{L}(P'') \\ & \Rightarrow & p_k = p'_k + p''_k \end{array}$$

$$\begin{array}{lrc} \textit{règles de produit} & & P \to P'P'' \\ & \Rightarrow & \mathcal{L}(P) = \mathcal{L}(P').\mathcal{L}(P'') \\ & \Rightarrow & p_k = \sum_{i=1}^{k-1} p'_i * p''_{k-i} \end{array}$$

$$\begin{array}{lrc} \textit{règles terminales} & & P \to t \\ & \Rightarrow & \mathcal{L}(P) = \{t\} \\ & \Rightarrow & p_k = \left\{ \begin{array}{lll} 1 & Si & k = 1 \\ 0 & Sinon & \end{array} \right. \end{array}$$

L'algorithme 1 permet d'engendrer aléatoirement et uniformément les mots issus d'un langage algébrique avec une probabilité de $\frac{1}{|\mathcal{L}(\mathcal{G})_n|}$.

Algorithme récursif pour la génération aléatoire uniforme de mots issus d'un langage algébrique **Génération** $(P \in \mathcal{N}, k \in \mathbb{N})$:
if $(P \to t) \in \delta$ **then**
  return t
end
else if $(P \to P'|P'') \in \delta$ **then**
  Tirer aléatoirement $x \in \mathbb{R}$ compris entre 0 et 1 exclu.
  if $x < \frac{p'_k}{p_k}$ **then**
    return Génération (P', k)
  end
  else
    return Génération (P'', k)
  end

  else if $(P \to P'P'') \in \delta$ **then**
    Choisir $k' \in [1, k-1]$ avec la probabilité $\frac{p'_{k'} p''_{k-k'}}{p_k}$
    return Génération (P', k').**Génération** $(P'', k-k')$
  end
end

Méthode par rejet

Soit $\mathcal{A}$ un ensemble dans lequel on veut tirer des éléments aléatoirement mais pour lequel il est difficile d'utiliser une approche constructive et $\mathcal{B}$ un sur-ensemble de $\mathcal{A}$ pour lequel on sait faire un tirage aléatoire. La méthode par rejet consiste à tirer des éléments de $\mathcal{B}$ et à rejeter les éléments qui ne sont pas dans $\mathcal{A}$.

La probabilité d'engendrer un élément $\omega \in \mathcal{A}$ de taille n est alors de :

$$P(\omega|\mathcal{A}_n) = P(\omega|\mathcal{B}_n)\frac{|\mathcal{B}_n|}{|\mathcal{A}_n|} = \frac{1}{|\mathcal{B}_n|}\frac{|\mathcal{B}_n|}{|\mathcal{A}_n|} = \frac{1}{|\mathcal{A}_n|}$$

La complexité de la méthode est :

$$(\alpha(n) + \beta(n))\frac{|\mathcal{B}|_n}{|\mathcal{A}_n|}$$

où $\alpha(n)$ est la complexité du test d'appartenance à $\mathcal{A}_n$, et $\beta(n)$ est la complexité de la génération dans $\mathcal{B}_n$. On voit donc que plus la classe $\mathcal{B}$ est "resserrée" autour de $\mathcal{A}$, moins la complexité sera importante.

Méthode de Boltzmann

Cette méthode consiste à engendrer les objets $\omega \in \mathcal{C}$ selon une probabilité $P_x(\omega) = \frac{x^{|\omega|}}{C(x)}$ où $C(x) = \sum_{n\geq 0} c_n x^n$ est la série génératrice de dénombrement de la classe $\mathcal{C}$ et x un paramètre analytique affectant la performance du générateur [36]. On engendre des objets jusqu'à ce que des objets de taille n soient engendrés pour les générateurs en taille exacte ou de taille $[n(1-\varepsilon), n(1+\varepsilon)]$ pour les générateurs en taille approchée. On remarque que chaque objet de taille n est engendré avec

la même probabilité ce qui garantit l'uniformité de la génération pour un n donné [36].

Générateur de Boltzmann pour le non terminal P **Génération** $P(x)$
if $(P \to t) \in \delta$ **then**
  return t
end
else if $(P \to P'|P'') \in \delta$ **then**
  Tirer aléatoirement $y \in \mathbb{R}$ compris entre 0 et 1 exclu.
  if $y < \frac{P'(x)}{P(x)}$ **then**
    return Génération $P'(x)$
  end
  else
    return Génération $(P''(x)$
  end

  else if $(P \to P'P'') \in \delta$ **then**
    return Génération (P', k').**Génération** $(P'', k - k')$
  end
end

Nous pouvons également engendrer les mots de façon non uniforme suivant une loi de probabilité donnée. On peut attacher des probabilités à ces langages de deux façons différentes. Soit nous nous attachons à maîtriser la fréquence d'utilisation des règles de production et nous parlons alors de grammaires stochastiques, soit nous cherchons à maîtriser la composition des mots engendrés et nous parlerons de grammaires pondérées. Nous allons maintenant présenter plus précisément ces notions et expliquer pourquoi nous allons choisir les grammaires pondérées.

3 Grammaires stochastiques et grammaires pondérées

Définition 27 (Grammaire non contextuelle stochastique)**.** *Une grammaire non contextuelle stochastique (GNCS) est une grammaire non contextuelle définie par* $(\Sigma, \mathcal{N}, S, \delta, \mathcal{P})$ *où* Σ *est l'alphabet terminal,* $\mathcal{N}$ *l'alphabet non terminal, S l'axiome,* δ *l'ensemble des règles de production et* $\mathcal{P}$ *l'ensemble des probabilités associées aux règles de production. Soit* $p_{C_i}^{s_{ij}}$ *la probabilité associée à la règle* $C_i \to s_{ij}, s_{ij} \in (\Sigma \cup \mathcal{N})^*$*, on a* $\sum_j p_{C_i \to s_{ij}} = 1$.

Les GNCS sont une généralisation des HMM aux langages non contextuels [37]. On peut dès lors engendrer aléatoirement des mots appartenant au langage associé à une grammaire algébrique. Si l'on considère la suite de dérivation $(\delta_1, \ldots, \delta_k)$ depuis l'axiome S jusqu'au mot ω, la probabilité $p(\omega|\mathcal{P})$ d'engendrer le mot sachant les probabilités de transition $\mathcal{P}$ ω est :

$$p(\omega|\mathcal{P}) = \prod_{1 \leq i \leq k} p_{\delta_i}$$

L'apprentissage des probabilités pour les GNCS se fait par estimation automatique à partir de mots issus du langage grâce à l'algorithme "Inside-Outside" [9, 66] de façon à reproduire la fréquence des mots observés.

Un autre modèle de génération aléatoire de mots par des grammaires non contextuelles a été développé par Denise *et al* [31, 30] qu'ils nomment grammaires pondérées. Ils cherchent alors à engendrer les mots de façon à reproduire els fréquences observées. Nous donnons d'abord une description de ces grammaires avant de les comparer aux grammaires stochastiques.

Définition 28 (Fonction de pondération)**.** *On appelle fonction de pondération* π *pour un alphabet* Σ *une fonction* $\pi : \Sigma \to \mathbb{R}$*. Une telle fonction est étendue multiplicativement sur tout mot de* Σ^* *de la façon suivante :*

$$\pi(\omega) = \prod_{c\in\omega} \pi(c)$$

Elle est enfin étendue additivement sur un langage de la façon suivante :

$$\pi(\mathcal{L}) = \sum_{\omega\in\mathcal{L}} \pi(\omega)$$

Définition 29 (Grammaire non contextuelle pondérée)**.** *Une grammaire non contextuelle pondérée est une grammaire non contextuelle définie par* $(\Sigma, \mathcal{N}, S, \delta, \pi)$ *où* Σ *est l'alphabet terminal,* $\mathcal{N}$ *l'alphabet non terminal, S l'axiome,* δ *l'ensemble des règles de production et* π *une fonctionde pondération définie de* $\Sigma \to \mathbb{R}$*.*

On note dès lors $p(\omega|\pi)$ la probabilité d'engendrer le mot ω dont chaque lettre x se voit affecter une pondération π_x. On a alors pour une longueur n fixée :

$$p(\omega|\pi) = \frac{\pi(\omega)}{\sum_{\omega'\in\mathcal{L}} \pi(\omega')}$$

On peut alors réaliser une génération aléatoire non uniforme de mots algébriques. On peut alors réaliser une génération aléatoire uniforme ou non de mots par la méthode récursive de la façon suivante :

Théorème 2. *Soit* A_π *l'adaptation de l'algorithme récursif 1 obtenu en remplaçant*

$$(P \to t) \Rightarrow p_k = \begin{cases} 1 & Si \qquad k = 1 \\ 0 & Sinon \end{cases}$$

par

$$(P \to t) \Rightarrow p_k = \begin{cases} \pi(t) & Si \qquad k = 1 \\ 0 & Sinon \end{cases}$$

Lors de la phase de dénombrement préliminaire, les C_{in} *deviennent alors les sommes des poids des séquences de taille n issues des* C_i*, c'est à dire :* $c_n = \sum_{\omega\in\mathcal{L}(C_i)_n} \pi(\omega)$*. Alors* A_π *engendre bien les mots de* $\mathcal{L}(G)_n$*, la longueur n étant fixée, avec les probabilités* $p(\omega|\pi) = \frac{\pi(\omega)}{\sum_{\omega'\in\mathcal{L}} \pi(\omega')}$*.*

Contrairement à la génération aléatoire par des grammaires pondérées, la génération aléatoire par GNCS ne nécessite pas d'étape de précalcul. Cette facilité est contre-balancée par la difficulté à contrôler la taille et la composition des mots engendrés. Weinberg et Nebel adaptent les GNCS de manière à n'engendrer que des mots de taille voulue [111] La méthode consiste en l'ajout d'un caractère terminal spécial permettant de terminer le mot lorsque l'on arrive à la taille voulue et à introduire une étape de précalcul de façon à contrôler la taille du mot engendré. Ils ne parviennent pas pour autant à contrôler la composition des mots que l'on souhaite engendrer.

Au contraire, les grammaires pondérées permettent un contrôle fin de la composition des mots engendrés tout en maintenant tout à fait naturellement la taille des mots constante. Le seul obstacle à cette méthode réside dans la difficulté à résoudre de façon exacte le calcul des pondérations de chaque lettre terminale à partir des fréquences observées. Cette difficulté peut être contournée par l'utilisation d'heuristiques qui permettent un calcul approché de ces pondérations [30] grâce au logiciel GRGFREQ.

C'est ce contrôle fin de la composition des mots engendrés qui nous fera retenir le modèle des grammaires pondérées. Nous utiliserons le logiciel GENRGENS développé par Ponty *et al* [30] afin d'exploiter nos modèles aléatoires.

4 Calcul asymptotique de coefficients de séries algébriques

Nous rappelons ici une méthode de calcul classique du comportement asymptotique des séries génératrices algébriques. Pour plus de développement, on pourra se référer à [42]. Nous disposons d'une équation polynomiale $F(z,S(z))$ où $S(z)$ est une série génératrice. Comme l'équation est polynomiale, $S(z)$ est une fonction algébrique. D'après le théorème de Cauchy, le comportement asymptotique de $S(z)$ vient de son comportement local en son point singulier.

Théorème 3.
$[z^n]S(z) = \frac{\alpha}{2\sqrt{\pi}}\omega^n n^{-3/2}(1+O(1/n))$, *avec* $\alpha = \sqrt{\frac{2\rho\partial F/\partial z|_{z=\rho,S=\sigma}}{\partial^2 F/\partial S^2|_{z=\rho,S=\sigma}}}$ *et* $\omega = 1/\rho$.

Démonstration. Rappelons d'abord le théorème de Cauchy, on a :

$$[z^n]S(z) = \frac{1}{2\pi i}\oint \frac{S(u)}{u^{n+1}}du$$

Nous obtenons :

$$[z^n]S(z) = \frac{1}{2\pi i}c_n \oint \frac{z^n}{z^{n+1}}dz$$

Nous obtenons ainsi l'expression de $[z^n]S(z)$. La contribution la plus grande à l'asymptotique est donnée par la singularité la plus proche de l'origine qui se trouve sur la branche partant de l'origine. La base de notre preuve est que sous de bonnes conditions :

$$[z^n]S(z) \approx O(\rho^{-n} + \Theta(n))$$

où ρ est la singularité et $||\Theta(n)|| = o(\rho^{-n})$ est exponentiellement négligeable.

Nous allons donc essayer de nous ramener à ce dernier cas et évaluer asymptotiquement le terme c_n. Pour cela, nous allons utiliserons le théorème de Newton-Puiseux qui affirme que dans le voisinage circulaire entourant la singularité dominante de $S(z)$, $S(z)$ admet un développement en série localement convergente. Il nous faut donc trouver cette singularité dominante.

Un point singulier z_0 de $S(z)$ est un point au niveau duquel $\partial F/\partial S = 0$. La singularité dominante est la singularité de plus petit module qui se trouve dans le prolongement de la courbe en partant de l'origine. On doit vérifier pour notre étude que la fonction peut admettre un prolongement analytique en partant de l'origine, c'est à dire que $\partial F/\partial z$ est défini et que $\partial F/\partial S \neq 0$. Posons le point $(z = \rho, S = \sigma)$ comme étant la singularité dominante.

D'après le théorème des fonctions implicites, z et $\sigma - S(z)$ sont du même ordre de grandeur. C'est à dire que $\partial z/\partial S = -\frac{\partial F/\partial S}{\partial F/\partial z}$ satisfait $\partial F/\partial z = 0$ au niveau de la singularité $(z = \rho, S = \sigma)$. On peut alors réaliser le développement de Taylor suivant :

$$z = \rho + \tfrac{1}{2}\tfrac{d^2z}{dS^2}(S-\sigma)^2 + \tfrac{1}{3!}\tfrac{d^3z}{dS^3}(S-\sigma)^3 + \ldots,$$

$$1 - \tfrac{z}{\rho} = -\tfrac{1}{2\rho}\tfrac{d^2z}{dS^2}(S-\sigma)^2 - \tfrac{1}{3!\rho}\tfrac{d^3z}{dS^3}(S-\sigma)^3 + \ldots,$$

Or

$$\frac{d^2z}{dS^2} = -\frac{\partial^2 F/\partial S^2}{\partial F/\partial z}$$

On suppose ici que $\frac{d^2z}{dS^2} < 0$ alors localement on a $S < \sigma$. On choisit la racine de $1 - \frac{z}{\rho}$ qui est positive pour z réel et inférieur à ρ. En posant $\beta_1 = -\sqrt{\frac{1}{2\rho}\frac{\partial^2 F/\partial S^2}{\partial F/\partial z}}$, on obtient le développement en série de Taylor :

$$\sqrt{1 - z/\rho} = \beta_1(S-\sigma) + \beta_2(S-\sigma)^2 + \ldots,$$

En inversant localement l'équation, on obtient :

$$S = \sigma - \sqrt{\frac{2\rho\partial F/\partial z|_{z=\rho,S=\sigma}}{\partial^2 F/\partial S^2|_{z=\rho,S=\sigma}}}\sqrt{1 - z/\rho} + O(1 - z/\rho) \quad .$$

Or on connait le développement en série entière de $\sqrt{1 - z/\rho}$

$$\sqrt{1 - z/\rho} = \Sigma_n \tbinom{n+\frac{1}{2}}{n} \rho^{-n} z^n$$

On a donc :

$$c_n = \tbinom{n+\frac{1}{2}}{n} \rho^{-n}$$

et

$$\tbinom{n+\frac{1}{2}}{n} \sim \tfrac{1}{2\sqrt{\pi}} n^{-3/2}(1 + O(1/n))$$

Il s'en suit que :

$$[z^n]\sqrt{1 - z/\rho} = \tfrac{1}{2\sqrt{\pi}}\rho^{-n} n^{-3/2}(1 + O(1/n)) :$$

$$[z^n]S(z) = \sqrt{\frac{2\rho\partial F/\partial z|_{z=\rho,S=\sigma}}{\partial^2 F/\partial S^2|_{z=\rho,S=\sigma}}}\frac{1}{2\sqrt{\pi}}\rho^{-n}n^{-3/2}(1+O(1/n)).$$

La solution est donc de la forme :

$$\frac{\alpha}{2\sqrt{\pi}}\omega^n n^{-3/2}(1+O(1/n)),$$

avec $\alpha = \sqrt{\frac{2\rho\partial F/\partial z|_{z=\rho,S=\sigma}}{\partial^2 F/\partial S^2|_{z=\rho,S=\sigma}}}$ et $\omega = 1/\rho$.

Le cas $\frac{d^2z}{dS^2} > 0$ se traite de façon symétrique.

Si $\frac{d^2z}{dS^2} = 0$ on reprend alors le raisonnement pour la première dérivée partielle non nulle [42].

□

Deuxième partie

Modèles de structures sans pseudonœud

Chapitre 3

Application de la génération aléatoire à la génération de bruit et au calcul de Z-valeurs de comparaison de structures.

Afin de déterminer si deux séquences sont homologues ou non, il nous faut évaluer la significativité du score de comparaison. Pour cela, on peut utiliser des grandeurs statistiques comme la Z-valeur. Cette grandeur nous permet d'évaluer la significativité d'un score en le comparant à un score obtenu par hasard, le hasard étant modélisé par une distribution bien choisie sur l'espace des séquences. Dans le cas des comparaisons de structures d'ARN, ces distributions ne sont pas connues. C'est pourquoi nous sommes amené à proposer ici un modèle de calcul empirique de la Z-valeur pour les alignements d'ARN. Ce calcul repose sur l'obtention de structures aléatoires d'ARN. Après avoir dressé un état de l'art concernant le calcul de la Z-valeur (section 1), nous allons proposer plusieurs modèles possibles de structures aléatoires basés sur des modèles de Markov ou des grammaires algébriques pondérées (section 2). Nous allons appliquer ces modèles dans deux contextes différents. Dans un premier temps, notre but sera de tenter de montrer que la génération aléatoire de structures d'ARN permet de fournir des sources de bruits plus pertinentes que des sources de bruits issues de séquences biologiques (section 3.2). Dans un deuxième temps, nous allons montrer que la génération aléatoire permet de calculer la Z-valeur de scores d'alignement de structures (section 4). Cette Z-valeur permet de mieux détecter si un ARN structuré appartient à une famille connue, en utilisant le critère du score.

1 Motivations de la génération aléatoire

Dans cette section, nous proposons quelques rappels concernant les notions d'alignement de structures ainsi que les avantages de l'utilisation de la Z-valeur sur le score brut dans le cas des séquences protéiques.

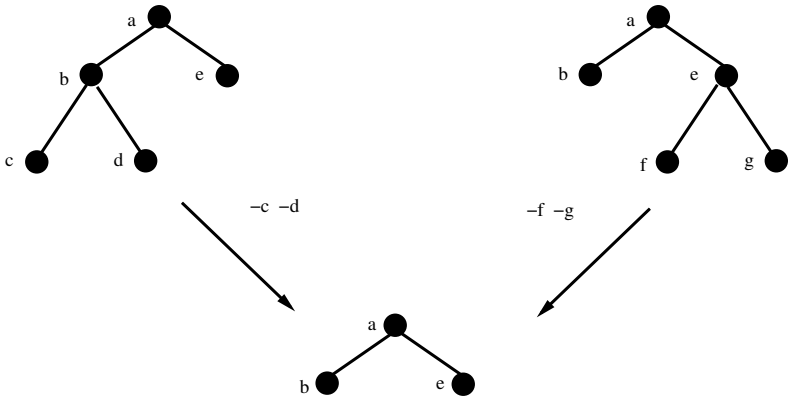


FIGURE 3.1 – ***Les opérations d'édition pour passer de l'arbre de gauche à l'arbre de droite nous conduisent à calculer le plus grand sous-arbre commun (au centre).***

1.1 Notion d'alignement et d'édition

En biologie, la similarité entre deux molécules traduit leur homologie. Cette similarité doit cependant être étayée par des mesures précises. C'est pourquoi nous utilisons la notion d'alignement. Nous avons vu dans la section 1.2 que les structures d'ARN peuvent être vues comme des arbres. Bien que d'autres méthodes soient également utilisées, nous présentons la similarité entre structures d'ARN en nous appuyant sur les mesures d'édition et d'alignement d'arbres.

La recherche de la distance d'édition entre deux arbres A et B consiste à rechercher l'ensemble d'opérations de coût minimum pour passer de l'arbre A au plus grand sous-arbre commun et de ce plus grand sous-arbre commun à l'arbre B (Figure 3.1). L'algorithme permettant de calculer cette distance est celui de Zhang et Shasha [115].

On peut aussi calculer la distance au plus petit sur-arbre commun aux deux structures (figure 3.2). On parle alors de distance d'alignement des deux arbres. Cette distance est calculée par l'algorithme de Jiang Wang et Zhang [60].

Nous allons maintenant présenter brièvement les principaux algorithmes de calcul de distance entre deux structures secondaires d'ARN. Dans ces cas, les opérations élémentaires permettant de calculer les distances tiennent également compte de la séquence nucléotidique.

1.2 Programmes d'alignement et d'édition de structures d'ARN

RNADISTANCE [58] implémente des opérations d'édition classiques sur les arbres. Cet algorithme ne prend pas en compte la séquence nucléotidique. RNAFORESTER [55] est un algorithme d'alignement d'arbres ordonnés local et global. Il utilise un codage spécial des arbres qui autorise à séparer les bases appariées sous certaines conditions. MIGAL [4] utilise une représentation des structures secondaires à plusieurs niveaux de granularité. Les structures sont codées par des arbres orientés ordonnés. La granularité va de la décomposition de la structure comme un ensemble de multiboucles jusqu'au niveau du nucléotide. Les scores employés

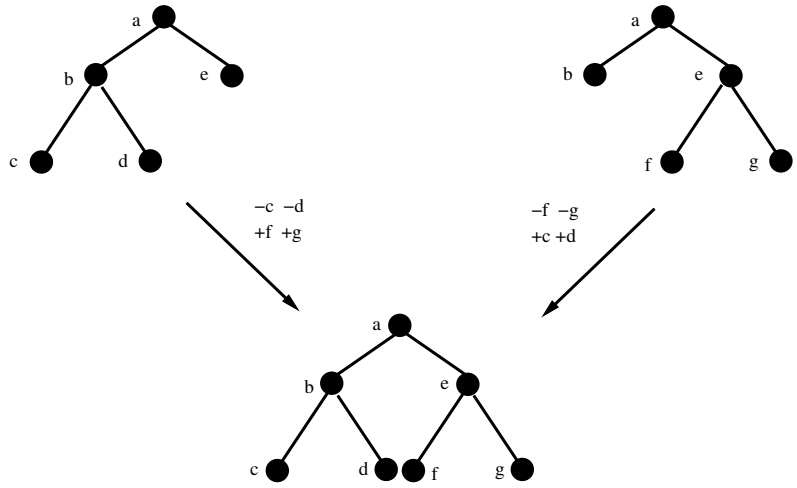


FIGURE 3.2 – ***Les opérations d'alignement de l'arbre de gauche avec l'arbre de droite nous conduit à calculer le plus petit sur-arbre commun au centre.***

sont des distances d'édition adaptées pour chaque niveau de granularité de la structure. TREEMATCHING [80] est basé sur une représentation de la structure sous forme d'arbre quotienté [38] qui est similaire à une structure composée d'arbres enracinés et ordonnés à des échelles différentes (nucléotidique et structurelle). La méthode repose sur la comparaison simultanée des deux niveaux : on calcule une distance d'édition entre les arbres quotientés à grande échelle en utilisant des distances d'édition entre les sous arbres à petite échelle. GARDENIA [12] et NESTEDALIGN [56] utilisent une représentation sous forme d'arcs annotés qui autorise des opérations complexes d'édition [11]. RNASTRAT [50] réalise les comparaisons en deux étapes. Tout d'abord, il compare les hélices des deux structures selon des opérations d'éditions complexes, puis il trouve une correspondance optimale entre les différentes hélices.

1.3 Score et Z-Score

Ces différents logiciels de comparaison renvoient un score de similarité. Dans [24], Comet *et al* utilisent la notion de Z-Score d'alignement, ils étudient plus particulièrement le Z-Score de similarité de séquences biologiques selon l'algorithme de Smith et Waterman [99]. La méthode consiste à comparer la première séquence avec un grand nombre de versions aléatoires de la seconde [68]. La simulation élimine le biais lié à la composition en résidus et à la taille de la séquence. On calcule $Z(A,B)$, le Z-Score d'alignement entre les séquences A et B de la façon suivante :

$$Z(A,B) = \frac{S(A,B)-m}{\sigma}$$

où m et σ, sont respectivement la moyenne empirique et la variance empirique de l'alignement entre A et N séquences aléatoires calculées à partir de B. Lorsque N augmente, la variance empirique σ diminue. Afin d'éliminer l'asymétrie entre $Z(A,B)$ et $Z(B,A)$, Comet *et al.* [24] recommandent de prendre le minimum des deux valeurs :

$$Z'(A,B) = min(Z(A,B), Z(B,A))$$

Comet *et al.* étudient également la loi de distribution de ces Z-Valeurs. Le score d'alignement de BLAST [5] suit un loi de distribution de Gumbel [64] :

$$P(X \geq s) = 1 - exp(-K.m.n.e^{-\lambda s})$$

où m et n sont les longueur des séquences alignées et K et λ sont calculées à partir des matrices de substitution et de la composition en résidus de la séquence.

la Z-Valeur suit cette loi de distribution dans le cas d'alignements entre séquences aléatoires. La distribution est alors :

$$P(Z \geq z) = 1 - exp(-e^{-(z-\zeta)/\theta})$$

avec :

$$\theta = (\frac{\sqrt{6}}{\pi})\sigma$$

$$\zeta = \mu - \gamma\theta$$

où $\gamma = 0.5772$ est la constante d'Euler.
Dans le cas d'alignement de séquences protéiques, la densité de probabilité suit une loi de Pareto :

$$f(z) = A.z^{-(1+\alpha)} \text{ si } z \geq 8$$
$$f(z) = 0 \text{ sinon}$$
$$\text{avec } \alpha \geq 0$$

Ici, les modèles aléatoires sont obtenus en réalisant des permutations des résidus dans la séquence initiale.

2 Différents modèles de structures aléatoires sans pseudonœud

Nous appliquons le calcul de la Z-valeur aux comparaisons de structures d'ARN. Contrairement aux algorithmes de comparaison de séquences, qu'elles soient nucléiques ou protéiques [99, 78, 5], il n'est pas intéressant d'engendrer une structure aléatoire par de simples permutations d'une structure de référence. En effet, nous souhaitons engendrer des structures de façon à garantir certaines propriétés (comme la conservation de la topologie ou la conservation du nombre moyen de tiges). De cette manière nous engendrons des structures aléatoires réalistes.

Pour construire des structures secondaires aléatoires, nous allons utiliser des modèles basés sur des grammaires. Nous présentons ces modèles dans les parties 2.1, 2.3 et 2.2. Nous obtenons un jeu de séquences et structures aléatoires à partir d'une séquence de référence et de sa structure à l'aide du pipeline développé dans la section 2.5. Ensuite nous utilisons ces résultats pour étudier les performances de différents algorithmes d'alignements de structures d'ARN dans la section 3.2, puis nous montrons en section 4 que le calcul de la Z-valeur permet d'améliorer l'identification de familles d'ARN par alignement.

2.1 Modèle de Markov d'ordre 1

Rivas et Eddy montrent que les structures d'ARN prédites à partir de séquences réelles ne différent pas des structures prédites à partir de séquences aléatoires [91]. Les séquences aléatoires sont engendrées à partir d'un modèle de Bernouilli de façon à respecter la fréquence en nucléotides d'ARN réels. Clote *et al* [22] montrent que si l'on considère la fréquence en di-nucléotides, les structures réelles ont une énergie plus basse que les structures obtenues par repliement de la séquence aléatoire. Dès lors, l'utilisation d'un modèle de Markov d'ordre 1 semble intéressant pour la génération de séquences aléatoires. Cette génération est réalisée à l'aide de GENRGENS. Les séquences obtenues sont alors repliées avec Mfold [117]. On ne garde que la première structure d'énergie optimale.

2.2 Modèle à topologie fixée

Définissons d'abord la notion de topologie ou "RNA shape" selon les travaux de Giegerich et *al* [48]

Définition 30 (Topologie d'une structure secondaire d'ARN)**.** *Considérons la transformation suivante :*

- *Chaque séquence* $\cdot^k$ *est remplacée par* $\cdot$
- *Chaque hélice* $({}^k\ldots)^k$ *est remplacée par* $(\ldots)$

Le mot obtenu correspond à la topologie de la structure d'ARN. Deux structures secondaires d'ARN sont topologiquement équivalentes si elles possèdent la même topologie.

Gevertz *et al* [46] constatent que le nombre de sous-structures (voir section 1.1) dans les aptamères sont peu nombreux et que leur nombre augmente avec la

taille des séquences. Ils constatent également que la distribution des topologies n'est pas uniforme dans les ARN *in vitro*. Ils montrent que la distribution du nombre de boucles dans des structures aléatoires de taille n suit une loi normale centrée en $n/20$. Ils montrent aussi que le nombre de multiboucles est faible dans les ARN aléatoires tandis qu'il est élevé dans les ARN fonctionnels. Ce travail nous a conduit à considérer un modèle de structures aléatoires qui conserverait la topologie de la structure de référence. Dans ce but, nous construisons la grammaire non contextuelle correspondant à la topologie de l'ARN de référence. Les terminaux correspondent aux bases et les non terminaux à la décomposition en tiges et en boucles. Nous souhaitons engendrer ces terminaux de façon à ce que leur fréquence soit la même que dans le modèle d'origine. Nous pouvons engendrer des structures garantissant ces fréquences observées dans le modèle de référence en calculant l'ensemble de pondérations π (Voir section 3).

Les règles de production sont les suivantes :
- Pour la production de tige :
  - $Tige \rightarrow (\ Tige\ ) \mid (\ StructureSuivante\ )$

- Pour les résidus non appariés :
  - $NonApp \rightarrow \cdot\ NonApp$

- Pour la production de boucles terminales :
  - $BoucleTerminale \rightarrow NonApp$

- Pour la production de renflement droit :
  - $RenflementDroit \rightarrow Tige\ NonApp$

- Pour la production de renflement gauche :
  - $RenflementGauche \rightarrow NonApp\ Tige$

- Pour la production de boucle interne :
  - $BoucleInterne \rightarrow NonApp\ Tige\ NonApp$

- Pour la production de multiboucle contenant k tiges :
  - $Multiboucle \rightarrow Tige_1 \ldots Tige_k$

  On peut intercaler des résidus non appariés entre les tiges. On obtient alors :
  - $Multiboucle \rightarrow Tige_1\ NonApp_1 \ldots Tige_k$

Considérons la structure représentée en figure 3.3) :

$$(((((((((\cdot\cdots\cdot(\cdot((\cdot\cdot(\cdots(((((\cdot\cdots\cdot\cdot)))))\cdots\cdot)\cdot\cdot)))\cdots\cdot)))))))))\cdot\cdot$$

La topologie correspondante est :

$$(\cdot(\cdot(\cdot(\cdot(\cdot)\cdot)\cdot))\cdot)\cdot$$

La grammaire associée sera :

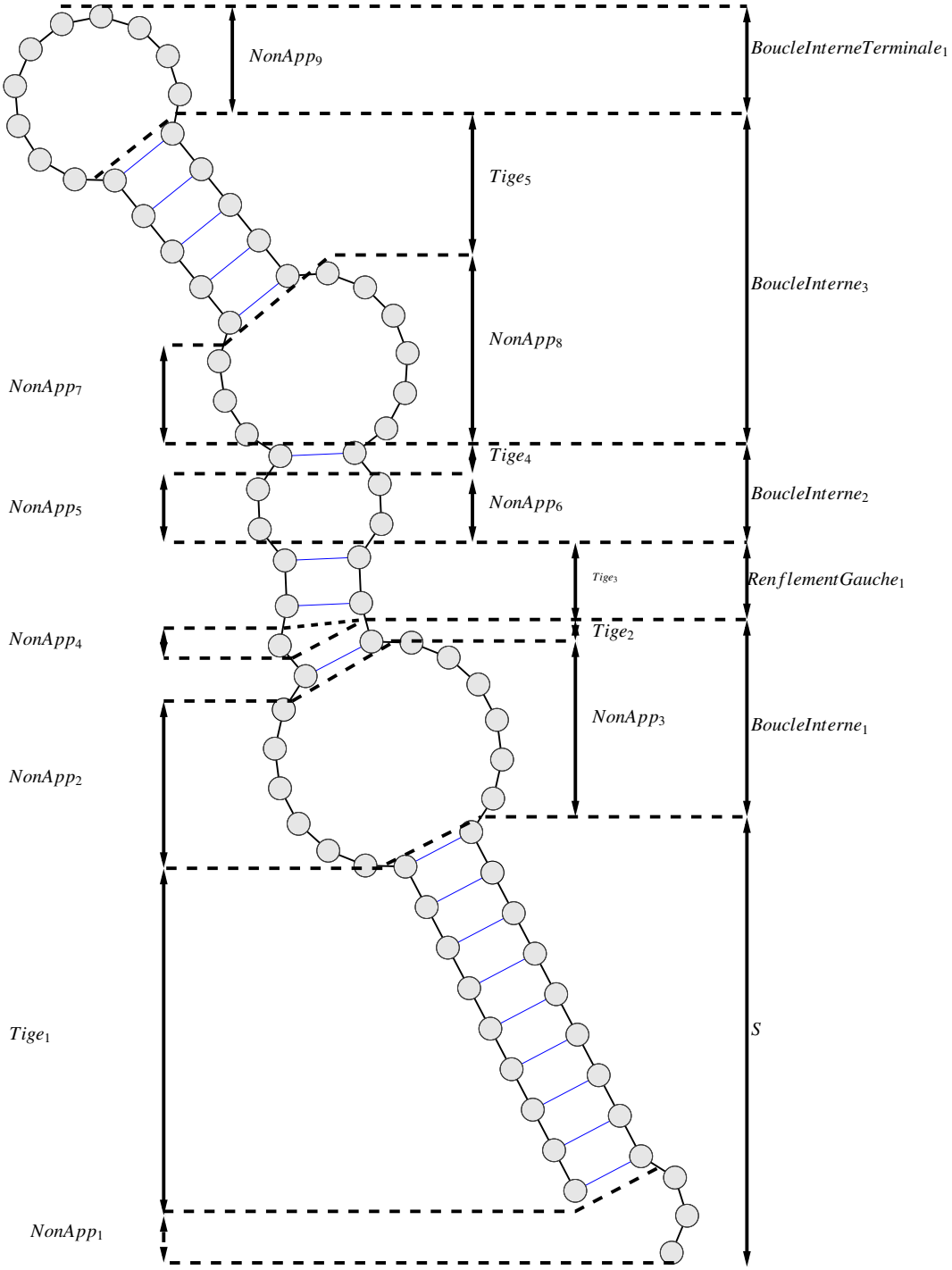


FIGURE 3.3 – ***Représentation d'une structure de micro ARN à l'aide de*** VARNA ***[29].***

$\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta, \pi)$
Avec :
$\Sigma = \{\cdot, (,)\}$

$\mathcal{N} = \{S, Tige_1, NonApp_1, BoucleInterne_1, NonApp_2, Tige_2, NonApp_3, RenflementGauche_1, NonApp_4, Tige_3, BoucleInterne_2, NonApp_5, Tige_4, NonApp_6, BoucleInterne_3, NonApp_7, Tige_5, NonApp_8, BoucleTerminale_1, NonApp_9\}$

$\delta =$

- $S \rightarrow Tige1\ NonApp_1$
- $NonApp_1 \rightarrow \cdot\ NonApp_1 \mid \cdot$
- $Tige_1 \rightarrow (\ Tige_1\ ) \mid (\ BoucleInterne_1\ )$
- $BoucleInterne_1 \rightarrow NonApp_2\ Tige_2\ NonApp_3$
- $NonApp_2 \rightarrow \cdot\ NonApp_2 \mid \cdot$
- $NonApp_3 \rightarrow \cdot\ NonApp_3 \mid \cdot$
- $Tige_2 \rightarrow (\ Tige_2\ ) \mid (\ RenflementGauche_1\ )$
- $RenflementGauche_1 \rightarrow NonApp_4\ Tige_3$
- $NonApp_4 \rightarrow \cdot\ NonApp_4 \mid \cdot$
- $Tige_3 \rightarrow (\ Tige_3\ ) \mid (\ BoucleInterne_2\ )$
- $BoucleInterne_2 \rightarrow NonApp_5\ Tige_4\ NonApp_6$
- $NonApp_5 \rightarrow \cdot\ NonApp_5 \mid \cdot$
- $NonApp_6 \rightarrow \cdot\ NonApp_6 \mid \cdot$
- $Tige_4 \rightarrow (\ Tige_4\ ) \mid (\ BoucleInterne_3\ )$
- $BoucleInterne_3 \rightarrow NonApp_7\ Tige_5\ NonApp_8$
- $NonApp_7 \rightarrow \cdot\ NonApp_7 \mid \cdot$
- $NonApp_8 \rightarrow \cdot\ NonApp_8 \mid \cdot$
- $Tige_5 \rightarrow (\ Tige_5\ ) \mid (\ BoucleTerminale_1\ )$
- $BoucleTerminale_1 \rightarrow NonApp_9$

– $NonApp_9 \rightarrow \cdot\ NonApp_9 \mid \cdot$

Les pondérations des symboles terminaux sont calculées à partir des fréquences observées, $\{\cdot,(,)\}$, sur la structure de référence grâce à GRGFREQS [30].

2.3 Modèle à nombre moyen de tiges fixé

Dans ce modèle on souhaite engendrer des structures possèdant en moyenne le même nombre d'hélices que le modèle de référence. Seuls ces paramètres sont pris en compte, la topologie est ici quelconque. Pour engendrer de telles structures, nous allons marquer par des crochets [...] les débuts des hélices au lieu des parenthèses classiques. Il nous suffit alors de maîtriser la fréquence des crochets lorsque l'on engendre les mots.

Par exemple :

$$(((((((((\cdot\cdots\cdots(\cdot((\cdot\cdot(\cdot\cdot\cdot(((((\cdot\cdots\cdots\cdot)))))\cdot\cdots\cdots\cdot)\cdot\cdot)))\cdot\cdots\cdots\cdot)))))))))\cdot\cdot\cdot$$

deviendra :

$$[(((((((((\cdot\cdots\cdots[\cdot[(\cdot\cdot[\cdot\cdot\cdot[((((\cdot\cdots\cdots\cdot))))]\cdot\cdots\cdots]\cdot\cdot)]]\cdot\cdots\cdots\cdot)))))))))]\cdot\cdot\cdot$$

On ignore ici la topologie de la structure, seul le nombre d'hélices, leur taille et le nombre de bases non appariées nous intéressent. Le nombre d'hélices correspond au nombre d'empilement de bases appariées dans la structure sans interruption par des bases non-appariées. De même que dans le modèle précédent, nous pouvons engendrer des structures garantissant les fréquences observées dans le modèle de référence en calculant l'ensemble de pondérations π (Voir section 3). On obtient alors la grammaire suivante par modification de la grammaire engendrant les mots de Motzkin de façon à distinguer le début des tiges : $\mathcal{G} = (\Sigma, \mathcal{N}, S, \delta, \pi)$ avec $\Sigma = \{\cdot,(,),[,]\}$ $\mathcal{N} = \{S, Tige\}$ $\delta =$

– $S \rightarrow [\ Tige\ ]\ S$

– $S \rightarrow \cdot\ S$

– $S \rightarrow \varepsilon$

– $Tige \rightarrow (\ Tige\ )\ S$

– $Tige \rightarrow \cdot\ S$

Ici, les symboles "[" et "]" correspondent aux deux parenthèses qui débutent une tige. En maitrisant la fréquence d'apparition de ces symboles, nous maitrisons le nombre de tiges engendrées. Nous souhaitons engendrer des structures dans lesquelle, le nombre d'appariements, de tiges et de résidus non appariés sont les mêmes en moyenne que dans la structure de référence. L'ensemble de pondérations π est calculé par GRGFREQS [30] à partir des fréquences objectif suivantes :

$$f(\text{"["}) = f(\text{"]"}) = \frac{nombredetiges}{taillesequence}$$
$$f(\text{"("}) = f(\text{")"}) = \frac{tailledestiges - nombredetige}{taillesequence}$$
$$f(\text{"}\cdot\text{"}) = \frac{nombredebasesnonappariees}{taillesequence}$$

2.4 Génération de structures et de séquences

Dans les deux modèles précédents, nous n'avons pas tenu compte des séquences. L'ajout des séquences sur la structure peut se faire de deux façons différentes :

- On augmente la taille de l'alphabet terminal en ajoutant une étiquette qui nous permet de savoir à quelle base les symboles $(,),[,],.$ sont associés. On a ainsi : $(_a,(_c,(_g,(_u,)_a,)_c,)_g,)_u,._a,._c,._g,._u,[_a,[_c,[_g,[_u,]_a,]_c,]_g,]_u$. On augmente aussi le nombre de règles de production de façon à tenir compte des ces nouveaux symboles et des appariements possibles.
- On engendre la structure sans la séquence et on la "décore" avec les bases en respectant les fréquences observées.

La première solution, bien que plus élégante, implique une augmentation significative du nombre de pondérations à calculer par GRGFREQ. En raison du temps de calcul que nécessite cette approche nous avons opté pour la seconde solution.

2.5 Pipeline pour la génération automatique de structures d'ARN aléatoires réalistes

Nous avons développé un pipeline de programmes qui permet d'automatiser la génération aléatoire de structures secondaires aléatoires selon les trois modèles présentés précédemment à partir d'une structure de référence.

Le pipeline est présenté dans la figure 3.4.

Nous partons d'un structure secondaire de référence qui est l'entrée de notre programme de décomposition de structures. Nous avons développé ce programme de décomposition en java. Le principe de l'algorithme de décomposition de structure en sous-structure est basé sur une décomposition en tiges boucles de la structure. Les sous-structures sont identifiées sur les critères indiqués en section 1.1. A partir d'une structure passée en parametre en entrée, le pipeline donne en sortie trois fichiers :

- Un fichier d'entrée pour GENRGENS permettant d'engendrer aléatoirement des séquences selon le modèle de Markov d'ordre 1.
- Un fichier d'entrée pour GRGFREQS permettant de calculer les pondérations afin d'engendrer aléatoirement les structures selon le modèle à nombre moyen de tiges fixées.
- Un fichier d'entrée pour GRGFREQS permettant de calculer les pondérations afin d'engendrer aléatoirement les structures selon le modèle à topologie fixée.

Les séquences engendrées selon le modèle markovien sont directement engendrées par GENRGENS puis repliées par MFOLD. Une fois les pondérations calculées, les structures correspondant aux modèles à topologie fixée et au modèle à nombre moyen de tiges conservé sont engendrées par GENRGENS.

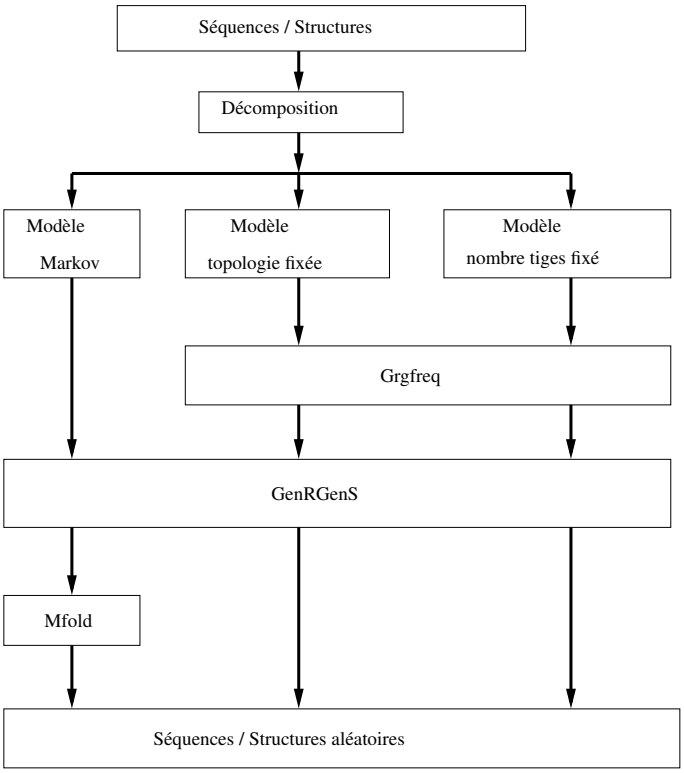


FIGURE 3.4 – ***Pipeline de programmes permettant d'obtenir des séquences structures aléatoires à partir d'une séquence et sa structure de référence.***

3 Structures aléatoires comme source de bruit.

Le travail présenté dans cette section a pour cadre le projet ANR Brasero. Il s'agit d'un projet multidisciplinaire réunissant biologistes et bioinformaticiens. Ce projet a, entre autres, pour but la constitution de données de benchmarks pour l'évaluation et la validation sur des réelles problématiques biologiques. Les séquences furent fournies par les biologistes Daniels Gautheret, Fabrice Leclerc et Yves D'Aubenton Carafa. Les développements bioinformatiques ont été principalement menés par Julien Allali et moi-même. Julien Allali et moi avons collaboré pour le développement du logiciel de comparaison des différentes méthodes de comparaisons de structures. Julien Allali a développé la calculateur de courbes ROC. Ce travail a fait l'objet d'une communication dans la conférence française de bioinformatique JOBIM [3].

3.1 Calcul des courbes ROC

Pour comparer le pouvoir séparateur des scores, on va réaliser les alignements entre 4 à 6 structures de référence et 100 structures d'ARN appartenant à la même famille que les ARN de référence. Ces alignements sont considérés comme positifs. On aligne 300 structures bruitées avec les structures de référence. Ces alignements sont considérés comme négatifs. Nous avons ainsi 25% de comparaisons positives contre 75% de négatives. L'utilisation de plusieurs séquences de référence nous permet de diminuer le risque lié à un biais éventuel.

Après avoir classé les scores de comparaison par ordre croissant, on considère un seuil S pour le score que l'on fait varier du score de comparaison le plus grand au score de comparaison le plus petit. Ce score S nous sert à classer les alignements. Si un score de comparaison est supérieur ou égal à S et est noté comme positif, on dit qu'il s'agit d'un vrai positif et on note VP. S'il est négatif on dit qu'il s'agit d'un faux positif noté FP. Si un score de comparaison est inférieur à S et qu'il est positif, on dira qu'il s'agit d'un faux négatif et on le note FN. Sinon il s'agit d'un vrai négatif et on le note VN.

L'ensemble du protocole est résumé dans la figure 3.5.

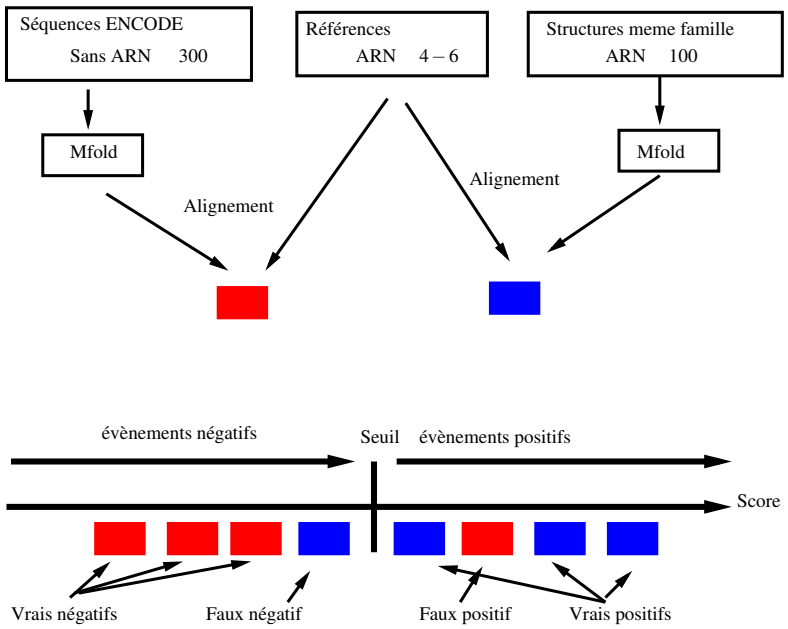


FIGURE 3.5 – ***Protocole expérimental pour le calcul des courbes ROC.***

On peut pour chaque valeur de S calculer la spécificité et la sensibilité :

$$sensibilite = \frac{VP}{VP+FN}$$
$$specificite = \frac{VN}{VN+FP}$$

On peut alors tracer la courbe ROC, spécificité = f(sensibilité) pour S allant du plus petit score au plus grand.

Meilleure est l'aire sous la courbe ROC pour une méthode d'alignement donnée, meilleur est le pouvoir séparateur du score.

3.2 Evaluation des différents programmes de comparaison.

Pour comparer les programmes de comparaison de structures, nous utilisons le protocole décrit en figure 3.5 et dans la section 3.1. Les séquences étudiées sont des tRNA [103]. Pour le bruit, nous utilisons des séquences de la même taille que l'ARN. Nos sources de bruit sont les suivantes :

– ENCODE [26]. En effet, il est admis que ENCODE ne contient pas de séquences correspondant à des gènes d'ARN non codant.

- Génomes viraux dans lesquelles il n'y a pas d'ARNt. Ces séquences sont les suivantes :
  - AF486073 : Acute bee paralysis virus isolat Poland 1, génome complet.
  - DQ447655 : Lake Victoria marburgvirus - Angola2005 strain Ang1386, génome complet.
  - EF157977 : European bat lyssavirus 2 isolat RV1333, génome complet.
  - EF452494 : Chikungunya virus strain TSI-GSD-218-VR1, génome complet.
  - NC001616 : Potato virus Y, génome complet.
  - NC003839 : Tomato ringspot virus RNA 2, séquence complète.
- Des séquences aléatoires engendrées selon le modèle à topologie fixe (structure et séquence). Ce modèle est celui qui nous permet d'engendrer les structures les plus réalistes. Il nous a donc paru naturel de l'utiliser afin de mettre en difficulté les logiciels d'alignements.

Les séquences extraites de ENCODE et les séquences virales ont été repliée par RNASUBOPT. Ce logiciel ne fournit pas seulement la structure de plus basse énergie mais aussi les structures sous-optimales. Dans le cas de cette expérience, les comparaisons ont été effectuées avec les sous-optimaux calculés et nous avons conservé celui qui permet d'obtenir le meilleur score lors de comparaison avec les structures de références.

Nous avons développé un logiciel en JAVA pour automatiser ce travail (figure 3.6)

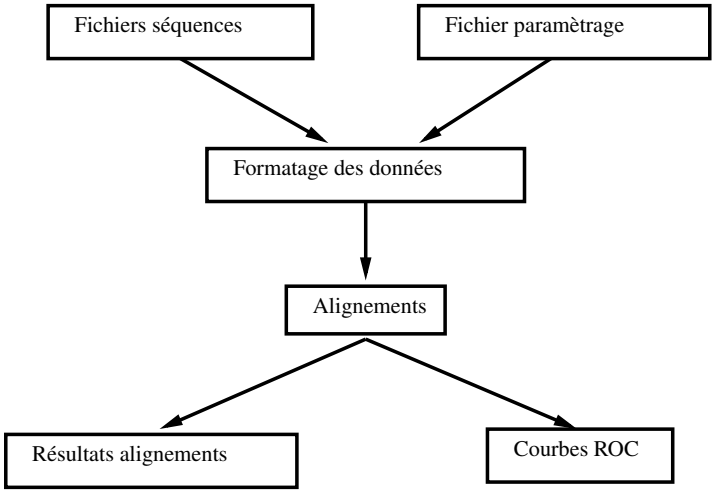


FIGURE 3.6 – ***Schéma de fonctionnement du programme de comparateur de programmes de comparaison.***

Nous avons utilisé les logiciels suivants pour réaliser ces comparaisons : RNAFORESTER [55], MIGAL [4], TREEMATCHING [80], GARDENIA [12], NESTEDALIGN [56], RNASTRAT [50], RNADISTANCE [58], BLAST [5].

BLAST n'étant pas conçu pour la recherche d'ARN par similarité de structure, il constitue pour nous un témoin négatif de l'expérience.

Les résultats obtenus pour les ARNt pour les bruits Viraux (figure 3.7), ENCODE (figure 3.8) et Modèle Topologie fixe (figure 3.9) sont présentés dans les

figures 3.7, 3.8 et 3.9. Les résultats sont résumés dans le tableau 3.1

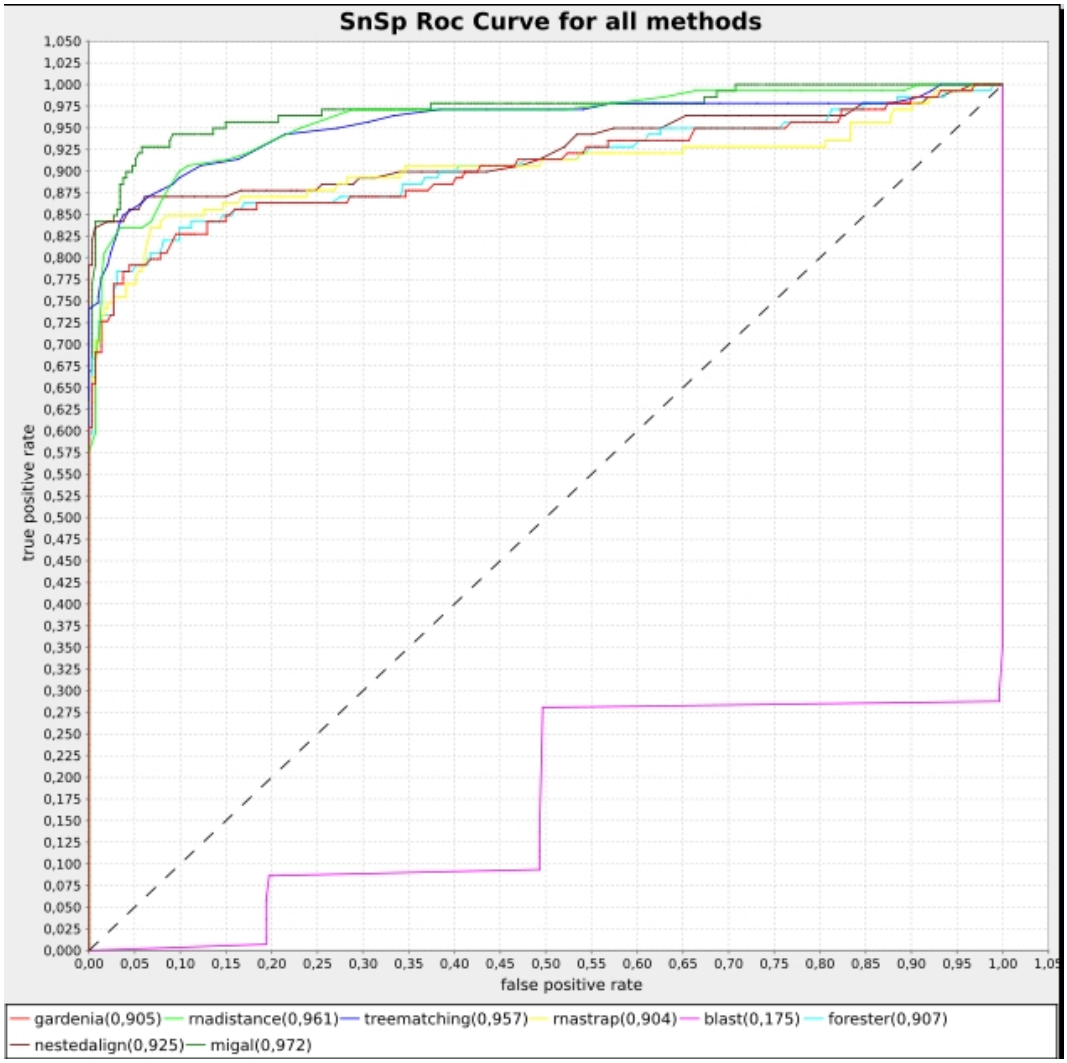


FIGURE 3.7 – ***Courbes ROC obtenues pour les différents logiciels de comparaison pour le modèle de bruit viral.***

La moyenne des écarts entre les aires de la première colonne et celle de la seconde colonne du tableau 3.1 est de 0.006 alors qu'elle est de 0.0228 entre la deuxième et la troisième et de 0.0265 entre la première et la troisième. On en déduit que les résultats obtenus pour les deux sources de bruit biologique (virales et ENCODE) sont similaires alors que les aires obtenues pour le bruit issus de nos structures aléatoires sont plus petites. L'utilisation de nos structures aléatoires comme source de bruit diminut le pouvoir séparateur des programmes d'alignement de structures par rapport aux structures d'origine biologique. Les variances des aires sous les courbes ROC sont sensiblement les mêmes pour les trois sources de bruit (9.6% pour ENCODE, 8.6% pour les séquences virales et 8.4% pour GenRGenS), on en déduit que le choix de la source de bruit ne permet pas de mieux distinguer les méthodes entre elles. Ces résultats sont encourageants pour généraliser l'utilisation des structures artificielles comme source de bruit pour les benchmarks à venir.

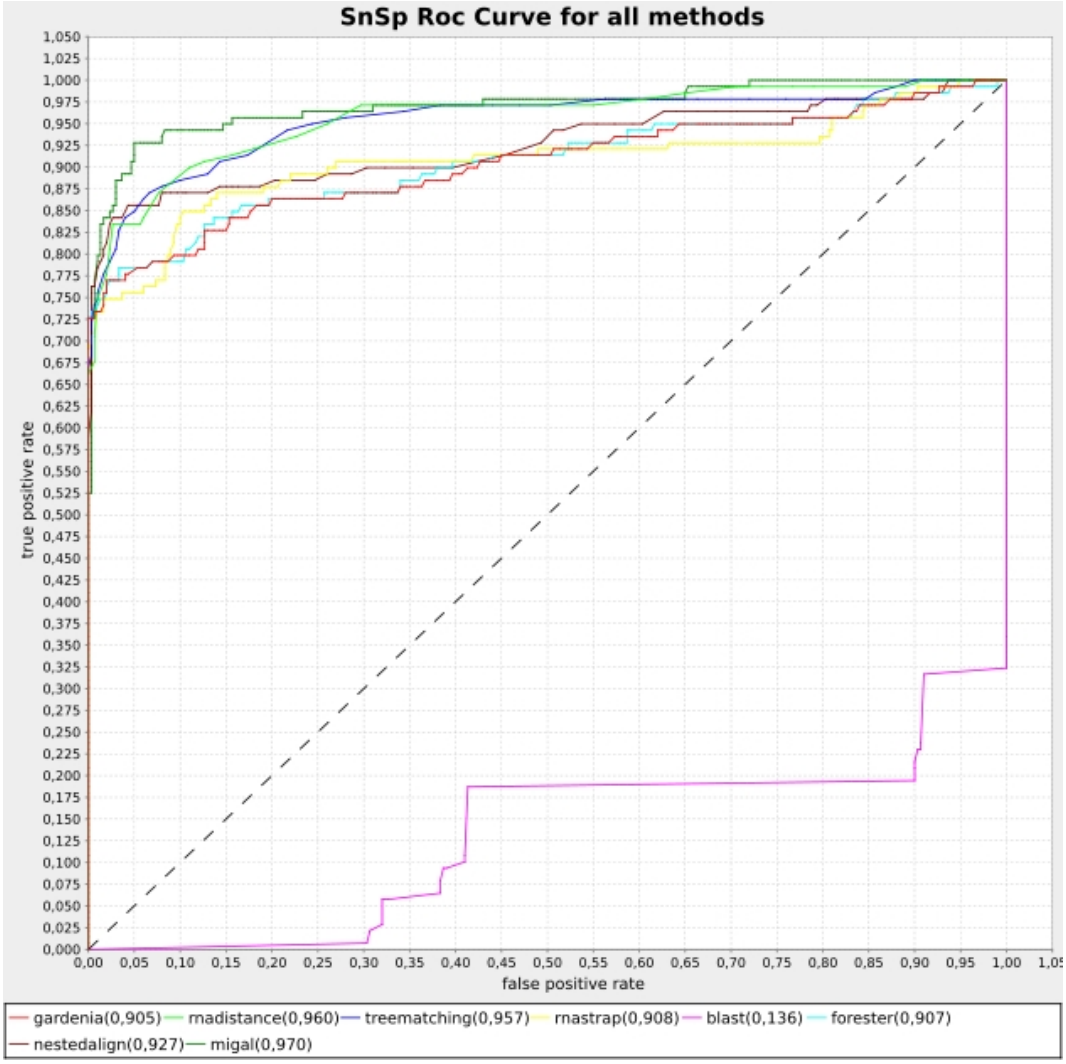


FIGURE 3.8 – ***Courbes ROC obtenues pour les différents logiciels de comparaison pour le modèle de bruit Encode.***

Méthode de comparaison	ENCODE	Viral	GENRGENS
GARDENIA	0.905	0.905	0.875
RNADISTANCE	0.960	0.961	0.934
TREEMATCHING	0.957	0.957	0.935
RNASTRAT	0.908	0.904	0.881
BLAST	0.136	0.175	0.169
RNAFORESTER	0.907	0.907	0.874
NESTEDALIGN	0.927	0.925	0.915
MIGAL	0.970	0.972	0.941

TABLE 3.1 – Tableau récapitulatif des aires sous les courbes ROC pour les différentes méthodes de comparaison selon la source de bruit utilisée.

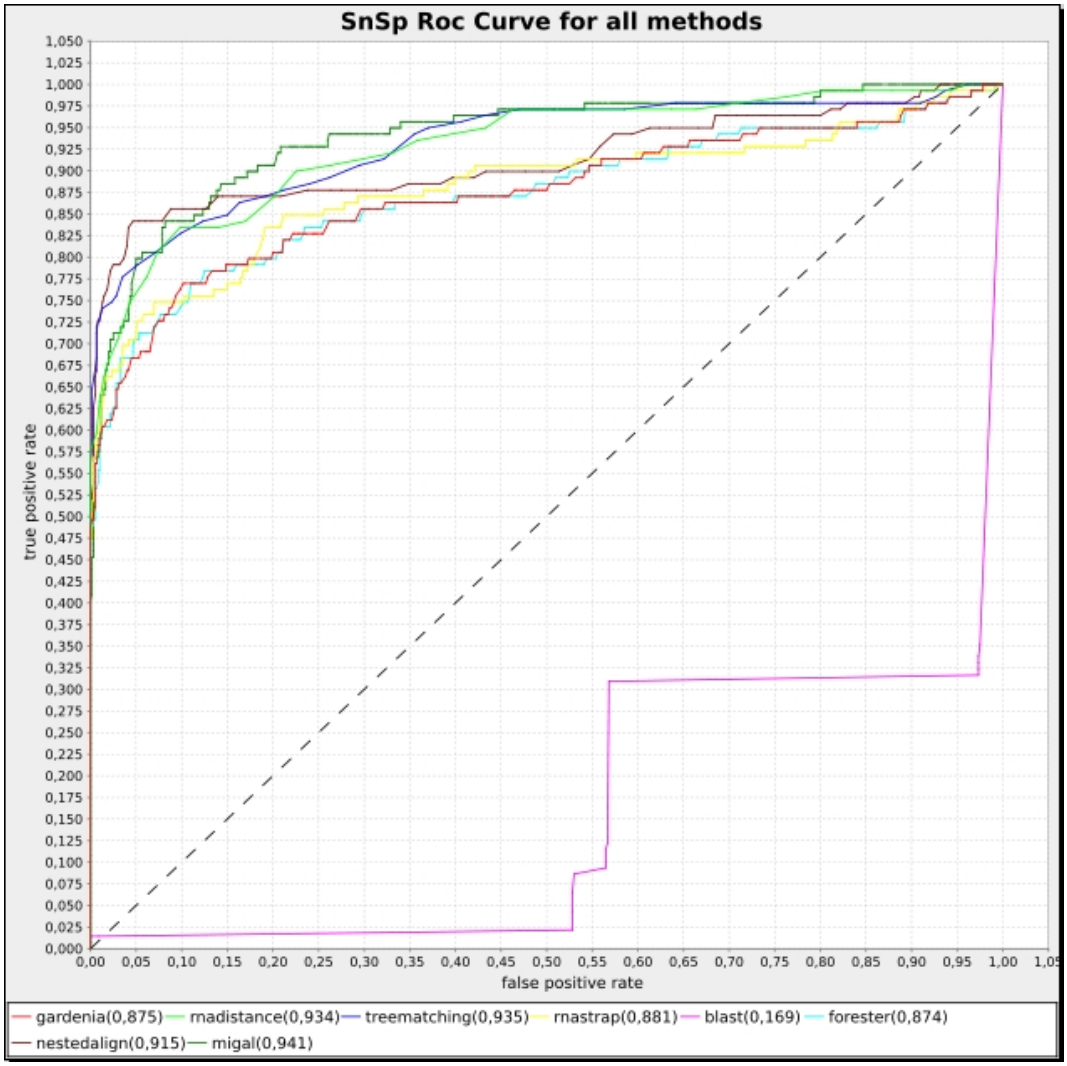


FIGURE 3.9 – ***Courbes ROC obtenues pour les différents logiciels de comparaison pour le modèle de bruit topologie fixe.***

4 Calculs de Z-valeurs

4.1 Calcul de la Z-valeur

D'une manière générale, le Z-score est calculé pour un score par rapport à une distribution de score. Il est calculé de la façon suivante :

$Z_{score} = \frac{score-\mu}{\sigma}$ où μ est la moyenne de la distribution des scores et σ son écart-type.

Nous nous intéressons au calcul du Z-score d'alignement de structures secondaires d'ARN. Plus précisément, nous souhaitons étudier le pouvoir discriminant du Z-score par rapport au score d'alignement brut. Par pouvoir discriminant nous entendons le fait d'attribuer un meilleur score d'alignement à l'alignement de deux ARN issus de la même famille et un score moins bon à l'alignement d'un ARN avec une source de bruit.

Nous calculons le Z-score de la façon suivante :

$$Z_{score}(A,B) = \frac{Score(A,B) - Moyenne(Score(A,Aleat(B)))}{Ecart(Score(A,Aleat(B)))}$$

où :

- A et B sont les deux structures à aligner.
- $Score(A,B)$ est le score de l'alignement de A avec B
- $Aleat(B)$ est une séquence aléatoire basée sur la séquence B
- $Moyenne(Score(A,Aleat(B)))$ est la moyenne des n alignements de A et n structures $Aleat(B)$
- $Ecart(Score(A,Aleat(B)))$ est l'écart moyen à la moyenne des n alignements de A et $Aleat(B)$

$n = 300$ est une valeur acceptable d'après [24]. Comme nous avons : $Score(A,Aleat(B)) \neq Score(Aleat(A),B)$, nous considérerons la Zvaleur :

$$Z_{valeur}(A,B) = Min \begin{cases} \frac{Score(A,B) - Moyenne(Score(A,Aleat(B)))}{Ecart(Score(A,Aleat(B)))} \\ \\ \frac{Score(A,B) - Moyenne(Score(Aleat(A),B))}{Ecart(Score(Aleat(A),B))} \end{cases}$$

On calculera la Z-valeur pour chaque modèle aléatoire considéré dans la section 2, c'est à dire :

- Modèle markovien d'ordre 1.
- Modèle à nombre de tiges fixées.
- Modèle à topologie fixée.

Nous allons donner les entrées de notre expérience. Nous souhaitons faire ici une preuve de concept. En effet, nous connaissons certaines propriétés de la Z-valeur dans le cas des alignement de séquences et nous souhaitons ici démontrer son intérêt dans le cadre des comparaisons de structures secondaires. Nous souhaitons donc nous affranchir de la séquence pour réaliser cette expérience. Pour cela nous utilisons RNADISTANCE qui compare les structures sans tenir compte des séquences.

Nous avons sélectionné les séquences de façon à étudier un échantillon représentatif des différentes familles d'ARN en termes de taille et de variabilité des structures. Nous avons sélectionné des familles de petite taille (ARNt [15, 103], ARNmi [49]) et de taille moyenne (RNAse P Eucaryote [14], SRP [6]). Nous avons également sélectionné des ARNsno [45]. Pour les ARNt, nous avons testé deux sources de repliement : soit un repliement par MFOLD [116] à partir des séquences fournies par Fabrice Leclerc [103], soit des structures obtenues sur la base de donnée Gutell [15]. Nous avons également testé des SRP issus de taxons différents (champignons et plantes) qui possèdent des structures différentes.

La source de bruit est consituée des séquences extraites de ENCODE [26] de la même taille que les 100 séquences de la famille de la référence. Ces séquences ne contiennent pas d'ARN. Les séquences sont extraites aléatoirement et repliées avec MFOLD [116].

4.2 Les Z-valeurs permettent une amélioration des aires sous les courbes ROC

Les aires sous la courbe ROC sont récapitulées dans le tableau 3.2

Famille	Score brut	Markov	Topologie fixe	Nombre moyen tiges fixe
miRNA	0.876	0.917	0.696	**0.989**
snoRNA	0.942	0.943	0.848	**0.995**
tRNA	0.893	0.900	0.825	**0.964**
tRNA Gutell	0.997	**0.998**	0.988	**0.998**
RNAseP eucaryote	0.874	**0.913**	0.803	0.815
SRP plante	0.975	**0.981**	0.904	0.970
SRP fungi	0.719	**0.928**	0.768	0.791

TABLE 3.2 – ***Tableau récapitulatif des aires sous les courbes ROC obtenues pour les différentes familles pour chaque modèle aléatoire.***

Les courbes ROC obtenues pour les ARNsno et les RNAseP sont indiquées dans les figures 3.10 et 3.11 (les autres courbes sont fournies dans l'annexe A) :

Les récapitulatifs de taille (et de variation de taille au sein de chaque famille) ainsi que le nombre de branchements maximum des boucles (ou degré) pour chaque famille se trouvent dans le tableau 3.3.

On peut regrouper ces résultats en deux groupes :

– Groupe 1 : miRNA, ARNsno, tRNA, tRNA Gutell
– Groupe 2 : RNAseP eucaryote, SRP fungi, SRP plante

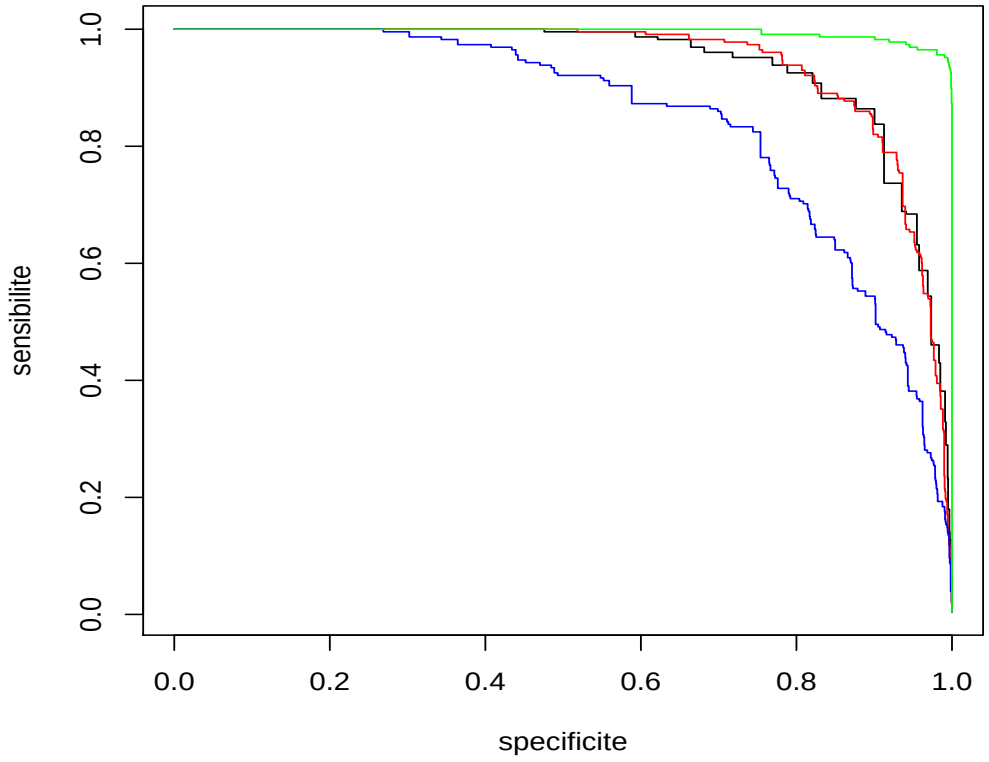


FIGURE 3.10 – ***Courbes ROC obtenues à partir des structures de ARNsno (Noire : Brut, Rouge : Markov ordre 1 , Bleue : Topologie fixe, Verte : Topologie nombre moyen de tiges fixe).***

Famille	Taille (variation en %)	Degré maximum moyen
miRNA	98.56 (23)	1.92
ARNsn	69.31 (13)	2.78
tRNA	71.43 (11)	3.52
tRNA Gutell	69.79 (18)	2.85
RNAseP eucaryote	309.64 (17)	3.55
SRP plante	276.16 (22)	2.99
SRP fungi	307.49 (40)	4.78

TABLE 3.3 – ***Tableau récapitulatif des tailles et du nombre de branchements maximum pour chaque famille.***

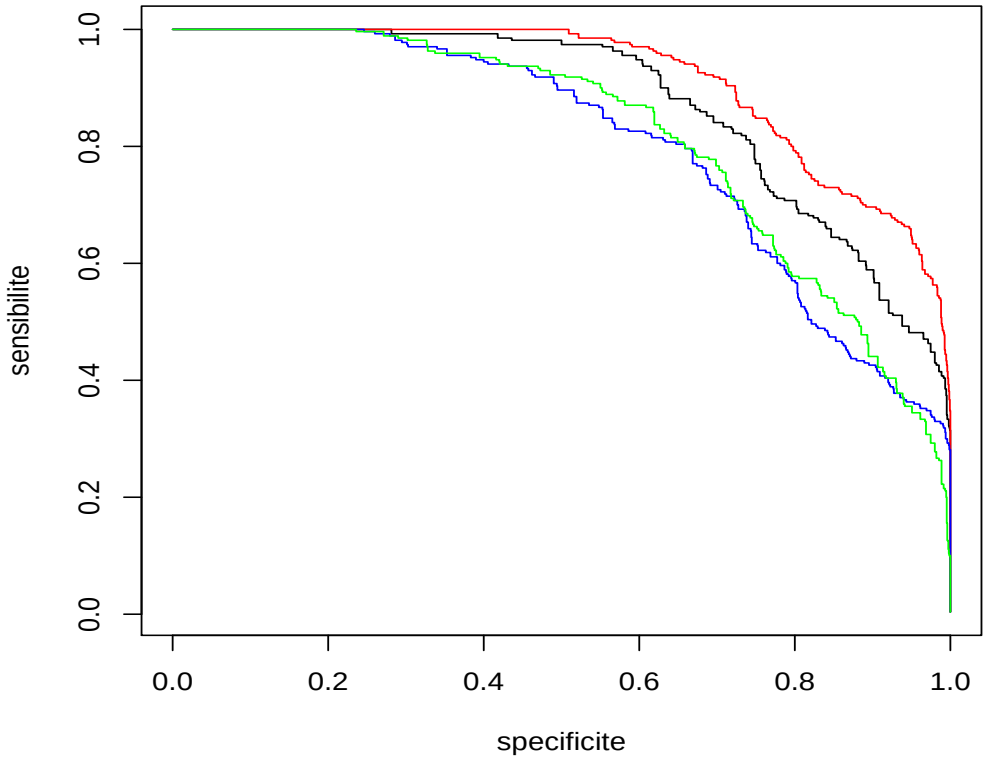


FIGURE 3.11 – ***Courbes ROC obtenues à partir des structures de RNAseP Eucaryotes (Noire : Brut, Rouge : Markov ordre 1 , Bleue : Topologie fixe, Verte : Topologie nombre moyen de tiges fixe.)***

L'analyse du tableau 3.3 nous permet d'affirmer que le groupe 1 rassemble des ARN de petite taille (moins de 100 résidus) et les deux des ARN de grande taille (plus de 200 résidus). Le degré maximum moyen ne semble pas influer sur les résultats.

On constate une amélioration globale des aires par l'utilisation des Z-valeurs (voir table 3.2). Dans le cas de la courbe 3.10, on remarque que le score pour le modèle markovien est proche du score brut, tandis que le score pour le modèle de topologie à nombre de tiges fixé leur est nettement supérieur et le score du modèle à topologie fixée est nettement inférieur à tous les autres. Dans le cas de la courbe 3.11, les courbes sont plus régulièrement espacées. On remarque cependant que le classement est vérifié tout au long du graphique.

L'utilisation des séquences aléatoires générées selon un modèle de Markov d'ordre 1 permet d'améliorer dans tous les cas l'aire sous la courbe ROC. Cela montre une forte contribution de la séquence. Détruire l'ordre de la séquence conduit à des structures similaires à celles obtenues par repliement de séquences sans ARN. On peut noter ici que RNADISTANCE ne tient pas compte de la séquence. Ce résultat suggère un biais de composition dans les séquences d'ARN. Un fort taux de GC pourrait par exemple expliquer ces résultats. Ce résultat suggère également que l'on devrait réitérer l'expérience avec un logiciel tenant compte de la séquence afin de voir la contribution exacte liée à la composition en résidus et celle liée simplement à la destruction générale de la structure non-contextuelle au sens strict dans la séquence.

Le modèle aléatoire engendrant le même nombre de tiges en moyenne améliore l'aire sous la courbe du groupe 1. Cela montre que la conservation de la topologie globale est la plus discriminante en matière de score d'alignement si l'on tient compte simplement de la structure et non de la séquence. Dans le cas des petits ARN, cette topologie moyenne est assez significative car il y a peu de possibilités d'engendrer des ARN ayant un faible nombre de tiges aléatoirement tout en conservant la taille globale des tiges. De fait, la modification de la topologie éloigne significativement la structure de la structure de base. Ce comportement est moins significatif pour les grands ARN pour lesquels la taille des tiges va largement contribuer au score de l'alignement.

L'utilisation du modèle aléatoire avec une topologie fixée détériore dans tous les cas l'aire sous la courbe ROC. On peut en conclure que les structures obtenues selon ce modèle ressemblent trop aux structures réelles tandis que de telles structures ne se retrouvent pas au hasard dans les séquences ne contenant pas d'ARN. Les structures engendrées par ce modèle sont trop proches des ARN servant à la calculer. Cela confirme le résultat de la section 3.2 dans laquelle nous avions montré que le modèle à topologie exacte permettait d'engendrer des structures plus réalistes que des aléatoires biologiques.

5 Conclusion et perspectives

Ces premiers résultats indiquent que le calcul et l'étude de la Z-valeur sont intéressants pour normaliser les scores de comparaison de structures d'ARN. On peut notamment remarquer que l'utilisation de la Z-valeur est particulièrement inté-

ressante pour déterminer à quelle famille appartient un ARN de petite taille. Dès lors il peut sembler intéressant de généraliser le calcul de la Z-valeur à toutes les méthodes d'alignement et examiner si les résultats obtenus avec RNADISTANCE sont confirmés. Un nombre plus important de familles pourrait également être testé (Intron, ARNr ...)

La Z-valeur dépend de plusieurs facteurs : l'algorithme utilisé pour le calcul du score, l'ensemble de scores pour les opérations unitaires, la distribution des scores de comparaison. L'étude de l'effet de ces différents paramètres sur la Z-valeur pourrait être intéressante.

Ce calcul de la Z-valeur nécessitant la moyenne empirique et l'écart moyen à la moyenne de la distribution des scores, il pourrait s'avérer intéressant de calculer cette distribution de scores empiriquement dans un premier temps. Ce travail pourrait etre obtenu en alignant entre elles un nombre suffisant de structures engendrées selon un modèle assez simple (par exemple l'ensemble des mots de Motzkin). On pourrait aussi étudier le comportement de cette distribution si l'on intégrait des contraintes supplémentaires dans le modèle comme une taille minimale des boucles terminales ou une longueur bornée pour les tiges. On pourrait alors étudier la distribution des alignements des structures d'ARN entre elles et les positionner dans cette distribution. Connaître l'influence des scores unitaires sur ces distributions pourrait également se révéler intéressant. On pourrait alors définir un critère objectif dans le choix de seuil ainsi que des valeurs pertinentes pour la p-valeur. L'idéal réside bien sûr dans la détermination théorique de cette distribution. Cette détermination demande une étude combinatoire fine du comportement des scores d'alignements. On pourrait par exemple la réaliser en étudiant systématiquement la distance d'édition entre tous les mots de Motzkin de même taille dans un premier temps, puis de taille différente par la suite.

En l'absence de la distribution des scores, on a vu que l'on pouvait utiliser des structures aléatoires. Il pourrait alors être intéressant de tester l'effet d'un plus grand nombre de modèles aléatoires (comme des modèles qui conserveraient le nombre de multiboucles, les boucles internes, les renflements ...) sur la courbe ROC des méthodes d'alignement. On pourrait aussi étudier l'influence du nombre d'alignements sur le calcul de S(x,Aleat(y)). Enfin, on sait que la structure secondaire n'est qu'une approximation de la structure réelle des ARN. Il peut être intéressant de prendre en compte dans la génération aléatoire des modèles plus complexes qui tiendraient compte de la présence des pseudonoeuds ou de liaisons tertiaires. Nous avons choisi d'explorer cette voie dans la partie qui va suivre en montrant que l'on peut engendrer aléatoirement des structures secondaires avec pseudonoeuds.

Troisième partie

Modèles de structures avec pseudonœuds

Chapitre 4

Introduction et état de l'art

Nous avons précédemment présenté les structures d'ARN d'un point de vue biologique et combinatoire. Nous savons que les structures secondaires sans pseudonœud pouvent être modélisées par des langages algébriques, descriptibles par des grammaires non contextuelles. Les structures secondaires comportant des pseudonœuds, quant à elles, ne sont pas descriptibles par des grammaires non contextuelles. De ce fait, il est plus difficile de les étudier. Pourtant, il existe un certain nombre d'algorithmes exacts de prédiction de structures secondaires avec pseudonœuds en temps polynomial. Alors que Condon *et al* établissent une hiérarchie stricte entre ces algorithmes, hiérarchie basée sur des relations d'inclusions entre leurs classes de structures, on constate qu'il n'existe pas de corrélation entre la hiérarchie de Condon *et al* et la hiérarchie de leur complexité.

Nous allons maintenant présenter les algorithmes de prédiction de structures d'ARN *ab initio* (section 1) et nous attarder plus particulièrement sur les algorithmes exacts afin de mieux comprendre les principes sur lesquels ils se basent pour la prédiction de pseudonœuds. Nous rappellerons les travaux d'énumération de structures ARN dans la section 2.

Puis, dans le chapitre 5, nous étudierons très précisément les espaces de prédiction des algorithmes en caractérisant les graphes de cohérence des structures qu'ils prédisent, cette description constituant une vision différente et complémentaire de celle de Condon *et al*. Nous nous intéresserons alors à la quantification de ces espaces de prédiction.

Enfin, dans le chapitre 6, nous évaluerons le compromis entre complexité en temps des algorithmes exacts de prédiction de structure secondaire *ab initio* et leur espace de prédiction. Nous montrerons comment, par des bijections bien choisies, nous pouvons dénombrer des classes pertinentes de structures pseudonouées grâce à une grammaire non contextuelle (Chapitre 6 section 1). Nous présenterons également d'autres résultats concernant l'énumération de structures d'ARN avec pseudonœuds par des méthodes bijectives (Chapitre 6 section 2), grâce auxquelles nous obtiendrons des formules closes pour le dénombrement de ces classes. Les grammaires non contextuelles obtenues précédemment nous permettront d'engendrer aléatoirement des structures avec certaines classes de pseudonœuds. Nous construirons un algorithme en temps linéaire qui permet d'engendrer ces structures pseudonouées de façon efficace, c'est à dire sans surcoût de complexité en temps par rapport à une génération de structures sans pseudonœud (Chapitre 6 section 3).

1 Algorithmes de prédiction de structure *ab initio*

1.1 Algorithmes exacts

Nous présentons ici les algorithmes de prédiction de structure ARN *ab initio* exacts. Ces algorithmes consistent à minimiser une fonction d'énergie par programmation dynamique (voir section 2 pour une présentation des modèles de repliement). Nous allons passer en revue les différents algorithmes, depuis ceux qui ont le plus faible pouvoir prédicteur jusqu'à ceux qui ont le plus fort.

Commençons par les algorithmes de prédiction de structures sans pseudonœud. L'algorithme de Nussinov et *al* prédit la structure secondaire sans pseudonœud qui maximise le nombre de paires de bases appariées. Il est basé sur la décomposition de la structure en bases non appariées, bases appariées dans une tige ou bifurcation. La complexité en temps de l'algorithme est en $O(n^3)$. L'algorithme de Zucker [116] prédit également les structures sans pseudonoeud mais utilise une décomposition de la structure en tiges et en boucles, ce qui permet de prendre en compte des paramètres thermodynamiques comme ceux de Turner [74]. L'un des programmes qui implante cet algorithme, nommé MFOLD, est l'un des plus utilisés pour calculer le repliement sans pseudonoeud [118]. La complexité en temps de l'algorithme est initialement en $O(n^4)$. Elle est diminuée en $O(n^3)$ grâce à l'utilisation d'un modèle simplifié pour le calcul de l'énergie des boucles internes [86]. Cette même simplification est applicable pour l'algorithme de calcul de la fonction de partition des structures sans pseudonoeuds de McCaskill [75].

Cao et Chen proposent un algorithme [16] pour le calcul de la fonction de partition selon un modèle thermodynamique différent de celui de McCaskill [75] et Dirks et Pierce [33]. Ils réalisent une représentation simplifiée des hélices sous forme de liens virtuels dont ils étudient toutes les conformations possibles afin de calculer l'entropie puis l'énergie libre associée. Les conformations des boucles sont étudiées par des marches auto-évitantes sur un réseau régulier. Grâce à ce modèle, ils prédisent la structure de plus basse énergie pour les pseudonoeuds de type H sans aucune boucle entre les hélices du pseudonoeud [17], puis avec des boucles ne contenant pas de sous-structures appariées dans les hélices du pseudonoeud [18]. Ce modèle plus contraint que celui de Reeder et Giegerich [87], mais dont le modèle d'énergie est le plus élaboré, est calculé en $O(n^6)$. Lyngsø et Pedersen proposent de calculer les structures ne comportant qu'un seul pseudonoeud de type H [72] (Voir section 1.1). Ce pseudonoeud ne doit pas être niché sous une autre structure. Ils proposent un algorithme en $O(n^5)$.

Les quatre algorithmes suivants s'intéressent aux pseudonoeuds de type H. Ces pseudonoeuds ne sont que des cas particuliers de pseudonoeuds simples. Dirks et Pierces [33] reprennent également le formalisme de Rivas et Eddy [89] pour construire leur algorithme. Ils calculent la fonction de partition (voir section 2) de structures avec pseudonoeuds de type H en étendant le modèle de McCaskill [75]. La complexité de leur algorithme est $O(n^5)$. L'algorithme de Reeder et Giegerich [87] s'intéresse à un modèle plus contraint de pseudonoeuds de type H. Dans leur modèle, il n'est pas possible d'insérer une sous-structure à l'intérieur des deux tiges du pseudonoeud. La figure 4.1, extraite de leur article [87], illustre cette contrainte. L'algorithme a une complexité en temps en $O(n^4)$.

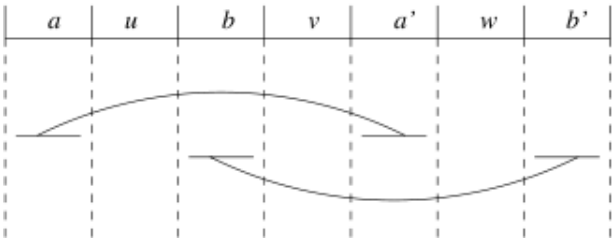


FIGURE 4.1 – ***Représentation des pseudonœuds prédit par PKnotRG [87]. Les deux hélices a, a' et b, b' ne sont interrompues par aucune sous-structure.***

L'algorithme de Uemura et *al* [106] prédit les structures secondaires avec pseudonoeuds décrites par une grammaire d'arbres adjoints moins générale que celle de Rivas et Eddy. Ces grammaires permettent d'insérer de nouvelles dérivations au sein d'un arbre de dérivations classique par l'utilisation de non terminaux particuliers. La complexité de l'algorithme est en $O(n^5)$. L'algorithme d'Akutsu [2] reformule l'algorithme d'Uemura et *al* de façon plus directe. Dans un premier temps, Akutsu présente un algorithme en $O(n^4)$ qui ne gère que les pseudonoeuds simples (Figure 4.2). Sa complexité passe en $O(n^5)$ dans sa version avec pseudonœuds quelconques. Le modèle sous-jacent reste une maximisation du nombre de paires de bases. Komina et *al* adaptent l'algorithme d'Akutsu pour le modèle de Turner [32].

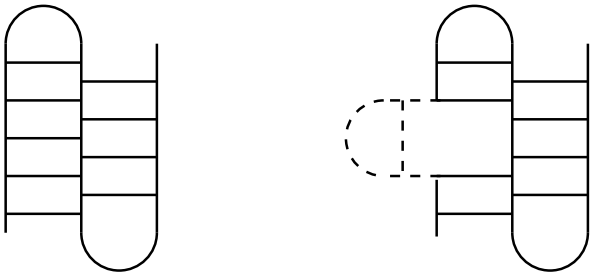


FIGURE 4.2 – ***Pseudonoeud simple à gauche et structure simplement pseudonouée à droite. Les lignes en pointillés marquent les sous-structures possibles.***

L'algorithme de Rivas et Eddy [89] ne permet pas de prédire toutes les structures avec pseudonoeuds. Les pseudonoeuds qu'il prédit sont décrits par une grammaire plus générale que les grammaires non contextuelles [90], grâce à un opérateur qui opére sur l'alphabet non terminal de la grammaire. Cette grammaire décrit un langage qui appartient à une classe intermédiaire entre les langages contextuels et non contextuels. L'opérateur est transposé en termes de programmation dynamique à l'aide d'une décomposition en structures qui ne se croisent pas. La figure 4.3 indique de quelle manière une telle décomposition est opérée. Leur algorithme a une complexité en temps en $O(n^6)$.

Enfin, le problème général du calcul de la structure qui minimise une fonction d'énergie a été prouvé NP-Complet [2, 72] y compris dans un modèle simple comme celui de Nussinov.

On récapitule ces différents algorithmes dans le tableau 4.1

Référence	Nom	Complexité	Pseudonoeuds	Modèle d'énergie	Fonction de partition
[79]	Nussinov	$O(n^3)$	Non	Maximisation des empilements	non
[116]	Zuker et Stiegler	$O(n^3)$	Non	Décomposition en tiges boucles	non
[75]	McCaskill	$O(n^4)$	Non	Décomposition en hélices et boucles	oui
[89]	Rivas et Eddy	$O(n^6)$	Type Rivas et Eddy	Décomposition en hélices et boucles	non
[106]	Uemura	$O(n^5)$	Simples et simplement pseudonouée	Maximisation des empilements	non
[72]	Lyngsø et Pedersen	$O(n^5)$	Un seul de type H non niché	Maximisation des empilements	non
[2]	Akutsu	$O(n^5)$	Simples et simplement pseudonouée	Maximisation des empilements	non
[33]	Dirks et Pierces	$O(n^5)$	Structure pseudonouée de type H	Décomposition en hélices et boucles	oui
[87]	Reeder et Giegerich	$O(n^4)$	Structure pseudonouée de type H restreint	Décomposition en hélices et boucles	non
[32]	Komina et *al*	$O(n^4)$	Simples	Minimisation de l'énergie, décomposition en hélices et boucles	non
[18]	Cao et Chen	$O(n^6)$	Structure pseudonouée de type H restreint	Minimisation de l'énergie, énergie libre liée aux conformations	oui

TABLE 4.1 – Tableau récapitulatif des différents algorithmes de prédiction de structure secondaire *ab initio* exacts.

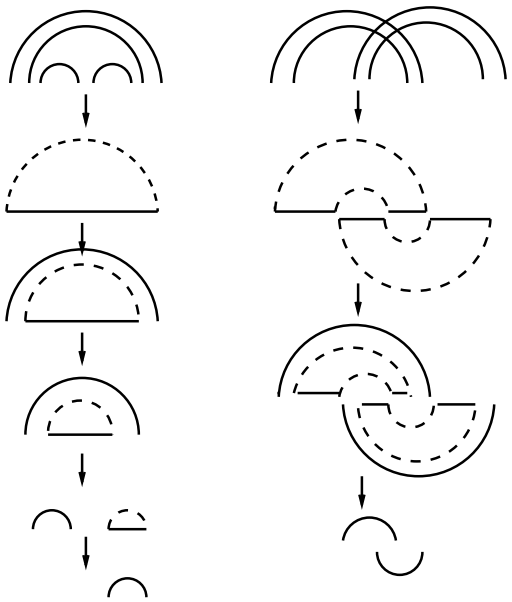


FIGURE 4.3 – ***A droite, la décomposition d'une structure en sous-structures sans pseudonoeud. Les traits pleins représentent les arcs tandis que les pointillés représentent les sous-structures potentielles. A gauche, une décomposition avec pseudonoeud. On peut remarquer que l'on utilise cette fois une représentation qui permet de dessiner les arcs sans croisement en insérant un "trou" dans la structure.***

1.2 Heuristiques

Bien que notre étude s'intéresse aux algorithmes exacts de prédiction de structure *ab initio*, nous présentons brievement les principales heuristiques de prédiction de structure ARN *ab initio*.

Les algorithmes génétiques cherchent une solution approchée au calcul de la fonction d'énergie. Les principaux algorithmes génétiques de prédiction de structure secondaire d'ARN sont l'algorithme de Gultayev [51] et celui de Batenburg [10]. L'algorithme de Gultayev repose sur un repliement hiérarchique qui tient compte de la cinétique du repliement. L'algorithme de Batenburg consiste en l'ajout d'hélices de façon à diminuer l'énergie libre de la structure.

FLEXSTEM [20] adopte une stratégie similaire en sélectionnant successivement des hélices candidates en recherchant des hélices maximales ; les solutions sont fusionnées et les conflits sont résolus en choisissant la solution qui minimise l'énergie de repliement. Abrahams *et al* [1], dans leur heuristique, font de même en minimisant l'énergie par descente locale. D'autres heuristiques utilisent ce principe de présélection des hélices. Le principe de HOTKNOTS [88] consiste à rechercher un ensemble de structures secondaires potentielles sans pseudonœud d'énergie minimum, puis d'y ajouter les structures résultant du calcul de l'énergie minimale. La procédure est réitérée sur les bases non appariées. Les pseudonœuds sont potentiellement ajoutés lors des deux dernières étapes.

ITERATED LOOP MATCHING (ILM) [93] recherche la structure secondaire d'énergie minimum à l'aide d'un algorithme à la Nussinov, puis la recherche des

pseudonœuds s'effectue en executant à nouveau l'algorithme sur les bases non appariées. En cela la méthode est similaire pour ILM et HOTKNOTS. KNOTSEEKER [100] recherche des appariements canoniques et garde ceux de plus petite énergie correspondant aux hélices les plus courtes. Elle forme ensuite des pseudonœuds de type H en appariant les boucles de façon à ne conserver que les structures les plus stables et de plus petite taille (c'est à dire que ces pseudonœuds ne sont formés que des hélices les plus stables et que l'énergie globale du pseudonœud est inférieure à celle des structures secondaires entrant en compétition). L'algorithme HFOLD développé par Jabbari *et al* [59] repose sur le repliement hiérarchique développé par Tinoco [104]. HFOLD prédit notamment les épingles à cheveux embrassées et possède une complexité en temps en $O(n^3)$. KINEFOLD [114, 113] est basé sur une modélisation de la cinétique de repliement de l'ARN par un processus de Monte-Carlo. MC-FOLD est le premier programme à prendre en compte les liaisons non canoniques [81]. Cette heuristique permet de calculer la structure tertiaire avec certains pseudonœuds. Il est toutefois difficile de décrire le type de pseudonœuds prédits par cette heuristique.

2 Dénombrement de structures d'ARN

Comme nous l'avons exposé dans la section 1.2, les structures secondaires d'sARN sans pseudonœud peuvent être vues comme des mots de Motzkin. La combinatoire de ces mots a été étudiée dans par Donaghey et Shapiro [34] et Stein et Waterman [102]. Les formes closes des formules de dénombrement des mots de Motzkin ainsi que leur construction sont présentées dans les travaux de Donaghey et Shapiro [34]. On peut également y trouver une bijection vers des arbres à boucles qui est une généralisation de la bijection des mots de Motzkin vers les arbres. On peut trouver la formule asymptotique du dénombrement de ces mots chez Stein et Waterman [102]. Waterman [110] compte le nombre de structures en se basant sur la décomposition récursive de la structure en tiges et en boucles et sur le nombre de branchements dans la structure. Vauchassade de Chaumont et Viennot [107] se basent sur la décomposition de Waterman pour donner la série génératrice qui dénombre ces structures. Hofacker *et al* [57] généralisent ces résultats en étudiant l'équivalent asymptotique de la formule de dénombrement et en discutant l'effet de la composition en nucléotides de la structure.

Des travaux sur le dénombrement tenant compte de modèles plus réalistes que les mots de Motzkin ont également été réalisés. En 2003, Nebel [77] étudie les statistiques d'apparition des sous-structures à partir des formules de dénombrement de Hofacker *et al* [57] ainsi que l'expression des différents moments de la formule de dénombrement. Clote *et al* [70] s'intéressent au dénombrement des RNAShape [48], développé par Giegerich *et al* ainsi qu'au calcul de leurs formules asymptotiques, ils montrent également l'existence d'une bijection entre les RNA shapes de taille $2n+2$ et les mots de Motzkin de taille n. De plus, Clote [21] s'intéresse à la combinatoire des structures d'ARN saturées, c'est à dire des structures secondaires dans lesquelles l'ajout d'un appariement entre bases non appariées introduirait un pseudonœud. Dans un autre travail, Clote *et al* [23] étudient également le nombre asymptotique de bases appariées dans des modèles aléatoires séquences d'ARN re-

pliée selon le modèle de Nussinov [79].

Les structures d'ARN pseudonouées ont également été étudiées du point de vue du dénombrement. Rodland [92] présente une façon de décomposer les structures d'ARN en composantes nouées et dénombre les pseudonœuds de type H. Vernizzi *et al* [108] s'intéressent au dénombrement des structures avec pseudonœuds selon le genre topologique de leur diagramme de corde en utilisant des arguments issus de la physique statistique. Enfin, le groupe de C. Reidys [61, 62, 63] étudie la formule de dénombrement des structures non croisantes d'ordre k, c'est à dire, les structures pour lesquelles il y a $k-1$ arcs mutuellement croisants ([62]), la forme asymptotique de ces formules ([63]), et réalise le dénombrement des structures non croisantes d'ordre k pour lesquelles les boucles font une taille ≥ 3 résidus et les tiges ont une longueur $\geq \sigma$.

Chapitre 5

Caractérisation des classes de Condon *et al* par des graphes de cohérence

Le problème de prédiction de structures secondaires avec pseudonœuds étant NP-Complet en toute généralité, les algorithmes polynomiaux ne peuvent prédire qu'un sous ensemble de structures possibles. En 2004, Condon *et al.* [25] classent les algorithmes exacts de prédiction de structure secondaire en comparant les ensembles de structures pouvant être prédites théoriquement par chacun d'eux. Nous appellons l'ensemble de structures que peut prédire un algorithme, "l'espace de prédiction de l'algorithme". Ils définissent ainsi cinq classes :

- *PKF* (PseudoKnot Free) pour les algorithmes de prédiction de structures sans pseudonoeud.
- *L&P* pour l'algorithme de prédiction de structures de Lyngsø et Pedersen.
- *D&P* pour l'algorithme de Dirks et Pierce.
- *A&U* pour la classe qui couvrent les pseudonœuds décrits par Uemura puis reprise par Akutsu.
- *R&E* pour l'algorithme de Rivas et Eddy.

Ils montrent que chaque classe de structures prédictibles par les différents algorithmes peut être caractérisée par des règles de ré-écriture [25] qui l'engendre. Nous allons tout d'abord donner la description de ces classes en utilisant les règles de Condon *et al* dans la section 1, puis nous allons utiliser la notion de graphe de cohérence [53] définie dans la section 1.1 afin de caractériser de façon différente et complémentaire les structures dans chacune de ces classes (section 2).

1 Classes et règles de ré-écriture de Condon et *al*

Définition 31 (Mot de Condon)**.** *Soit une structure secondaire $\mathcal{S}$ dont on supprime toutes les bases non appariées. On nomme distinctement chaque arc $(i, j) \in \mathcal{S}$ par une lettre $u \in \mathcal{A}$ où $\mathcal{A}$ est un alphabet donné. On nomme également u les pieds droite et gauche de l'arc u.*

On appelle mot de Condon le mot formé par la séquence des pieds des arcs lue de gauche à droite (Figure 5.1).

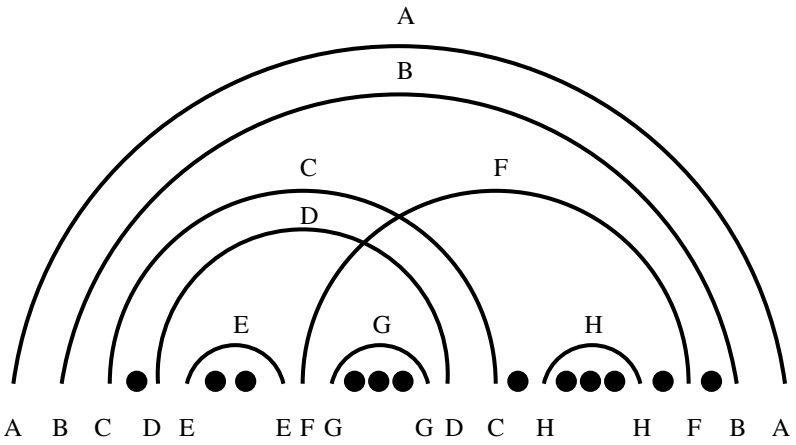


FIGURE 5.1 – ***Codage de la structure secondaire (en haut) par un mot de Condon (en bas). Chaque arc est nommé. Les pieds sont nommés de la même manière que l'arc. Les bases non appariées sont ignorées.***

Définition 32 (Classe de Condon)**.** *Une classe de Condon est l'ensemble des mots de Condon qui peuvent être réduits au mot vide par l'application successive d'un jeu de règles de ré-écriture caractéristiques d'un algorithme de prédiction de structures secondaires ab initio.*

Nous allons maintenant caractériser les différentes règles de ré-écriture pour chacune des classes de Condon. Pour chaque règle de ré-écriture, on donne dans la partie gauche les caractères sur lesquels vont s'appliquer la règle, on note en pointillés une séquence de caractères possiblement vide. Dans la partie droite on donne le résultat à l'issue de l'application de la règle de ré-écriture.

Définition 33 (Règles de ré-écriture de la classe PKF)**.** *Pour* $a, b \in \mathcal{A}$

1. $\ldots aa \ldots \longrightarrow \ldots \varepsilon \ldots$,
2. $\ldots ab \ldots ba \ldots \longrightarrow \ldots b \ldots b \ldots$,

La figure 5.2 nous donne un exemple de structure PKF, Le mot correspondant est : ABCCDDBAEE

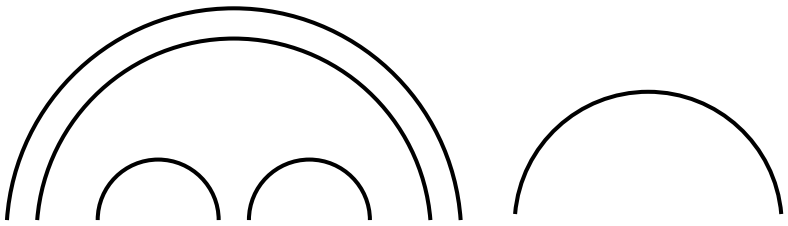


FIGURE 5.2 – ***Structure secondaire sans pseudonoeud,*** PKF***. Le mot de Condon correspondant est ABCCDDBAEE.***

Voici un déroulement de l'application des règles de ré-écriture sur le mot ABCCDDBAEE :

– Par application de la règle 1 sur E on obtient : ABCCDDBA

– Par application de la règle 2 sur B on obtient : BCCDDB
– Par application de la règle 1 sur D on obtient : ACCA
– Par application de la règle 2 sur C on obtient : CC
– Par application de la règle 1 sur C on obtient : ε

Définition 34 (Règles de ré-écriture de la classe *L&P*)**.** *Pour* $a, b \in \mathcal{A}$
1. $\ldots aa \ldots \longrightarrow \varepsilon$,
2. $\ldots ab \ldots ba \ldots \longrightarrow \ldots b \ldots b \ldots$,
3.1. $abab \longrightarrow \varepsilon$.

La figure 5.3 nous donne un exemple de structure *L&P*, Le mot correspondant est : ABCCDDBAEFGHHEGF

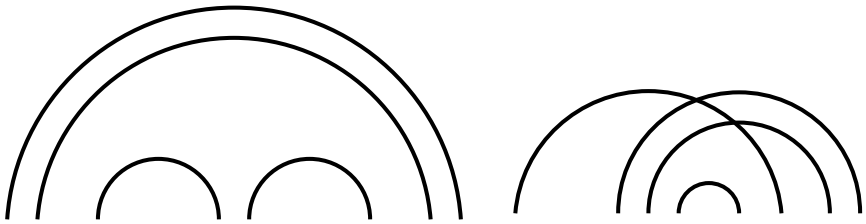


FIGURE 5.3 – ***Structure secondaire de la classe L&P. Le mot de Condon correspondant est ABCCDDBAEFGHHEGF.***

Voici un déroulement de l'application des règles de ré-écriture sur le mot ABCCDDBAEFGHHEGF :
– Par application de la règle 1 sur C, D puis H on obtient : ABBAEFGEGF
– Par application de la règle 2 sur B puis G on obtient : BBEGEG
– Par application de la règle 1 sur B on obtient : EGEG
– Par application de la règle 3.1 sur EGEG on obtient : ε
On remarque que la règle 3.1 ne peut être utilisée sur EGEG avant que les autres lettres n'aient été supprimées.

Définition 35 (Règles de ré-écriture de la classe *D&P*)**.** *Pour* $a, b \in \mathcal{A}$
1. $\ldots aa \ldots \longrightarrow \ldots \varepsilon \ldots$,
2. $\ldots ab \ldots ba \ldots \longrightarrow \ldots b \ldots b \ldots$,
3.2. $\ldots abab \ldots \longrightarrow \ldots \varepsilon \ldots$.

La figure 5.3 nous donne un exemple de structure *D&P*, Le mot correspondant est : ABCCDDBAEFGHHEGFIJIJ

Voici un déroulement de l'application des règles de ré-écriture sur le mot ABCCDDBAEFGHHEGFIJIJ :
– Par application de la règle 1 sur C, D puis H on obtient : ABBAEFGEGFIJIJ
– Par application de la règle 2 sur B puis G on obtient : BBEGEGIJIJ
– Par application de la règle 1 sur B on obtient : EGEGIJIJ
– Par application de la règle 3.2 sur EGEG puis IJIJ on obtient : ε

FIGURE 5.4 – ***Structure secondaire de la classe D&P. Le mot de Condon correspondant est ABCCDDBAEFGHHEGFIJIJ.***

On remarque que contrairement à la règle 3.1 de la classe *L&P*, la règle 3.2 peut être appliquée même si le mot *abab* n'est pas seul dans la séquence.

Définition 36 (Règles de ré-écriture de la classe *A&U*)**.** *Pour* $a,b \in \mathcal{A}$

1. $\ldots aa \ldots \longrightarrow \ldots \varepsilon \ldots$,
2. $\ldots ab \ldots ba \ldots \longrightarrow \ldots b \ldots b \ldots$,
3.3. $\ldots a \ldots bab \ldots \longrightarrow \ldots a \ldots a \ldots$ *Si et seulement si une autre règle peut être appliquée en partant de a*

La figure 5.5 nous donne un exemple de structure *A&U*, Le mot correspondant est : ABCDCEBFAFEGHGHD

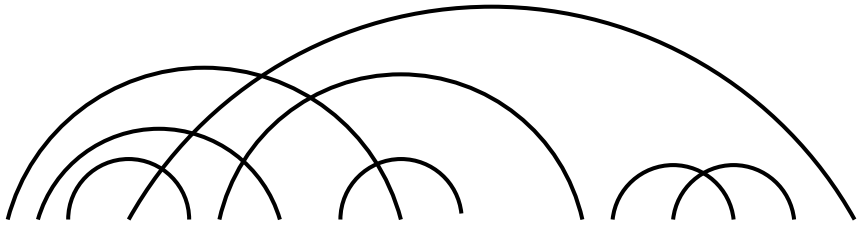


FIGURE 5.5 – ***Structure secondaire de la classe A&U. Le mot de Condon correspondant est ABCDCEBFAFEGHGHD.***

Voici un déroulement de l'application des règles de ré-écriture sur le mot ABCDCEBFAFEGHGHD :

– Par application de la règle 3.3 sur H : ABCDCEBFAFEGGD
– Par application de la règle 1 sur G : ABCDCEBFAFED

On remarque que comme indiqué en note de la règle 3.3, tant que G n'était pas supprimé, il était impossible d'utiliser une règle sur une autre lettre.

– Par application de la règle 3.3 sur F : ABCDCEBAED
– Par application de la règle 2 sur B : BCDCEBED
– Par application de la règle 3.3 sur E : BCDCBD
– Par application de la règle 2 sur C : CDCD
– Par application de la règle 3.3 sur D : CC
– Par application de la règle 1 sur C : ε

Définition 37 (Règles de ré-écriture de la classe *R&E*)**.** *Pour* $a,b \in \mathcal{A}$

1. $aa \longrightarrow \varepsilon$,

2. $ab \ldots ba \longrightarrow b \ldots b$,
3.4. $aba \longrightarrow b$,
4. $ab \ldots ab \longrightarrow b \ldots b$.

La figure 5.6 nous donne un exemple de structure *R&E*, Le mot correspondant est : ABCDACBD

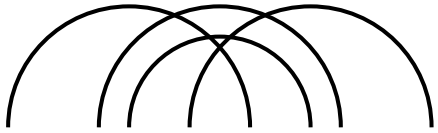


FIGURE 5.6 – ***Structure secondaire de la classe R&E. Le mot de Condon correspondant est ABCDACBD.***

Voici un déroulement de l'application des règles de ré-écriture sur le mot ABCDACBD :

– Par application de la règle 2 sur C on obtient : ACDACD
– Par application de la règle 4 sur D on obtient : ADAD
– Par application de la règle 3.4 sur A on obtient : DD
– Par application de la règle 1 sur D on obtient : ε

On remarque que plusieurs déroulements des règles sont possibles pour arriver au mot vide en partant d'un même mot de Condon.

Condon *et al* montrent grâce à ces règles de ré-écriture que l'on a la relation suivante :

$$PKF \subset L\&P \subset D\&P \subset A\&U \subset R\&E$$

Ils publient également un algorithme en temps linéaire pour reconnaître les séquences de *PKF*, *L&P*, *D&P* et *R&E* [25] et *A&U* en 2007 [85]. L'algorithme de reconnaissance en temps linéaire pour *A&U* concerne les pseudonœuds simples et non les structures simplement pseudonouées.

A partir de ces règles de ré-écriture, nous allons caractériser les graphes de cohérence de ces différentes classes.

2 Caractérisation des classes par des graphes de cohérence.

Le système de ré-écriture de Condon *et al* permet de décider si une structure appartient à une classe de structures. Nous proposons une vision différente et complémentaire de ces classes. Pour cela nous les caractérisons par leur graphe de cohérence. Par souci de clarté, nous donnons à nouveau la définition de graphe de cohérence.

Définition 38 (Graphe de cohérence [53])**.** *Le graphe de cohérence d'une structure secondaire d'ARN est le graphe qui associe à chaque appariement (i,j) un sommet, et tel que les sommets associés à (i,j) et (k,l) sont connectés si et seulement si $cross((i,j),(k,l)) = vrai$.*

2.1 La classe des structures sans pseudonœuds (PKF)

Proposition 1. *Le graphe de cohérence des structures de la classe PKF ne contient que des sommets isolés.*

Démonstration. Les structures de la classe PKF ne contiennent aucun arc croisant. De fait leur graphe de cohérence est un ensemble de sommets isolés.

□

2.2 La classe de Lyngsø et Pedersen ($L\&P$)

Théorème 4. *Les structures de la classe L&P contiennent au plus un pseudonœud de type H. Ce pseudonœud n'est pas imbriqué sous un arc.*

Nous allons démontrer ce théorème grâce aux deux lemmes suivants.

Lemme 1. *Un pseudonoeud de la classe L&P est un pseudonoeud de type H.*

Démonstration. Prenons une structure $\mathcal{S}$ de la classe *L&P*, $\mathcal{G} = (V,E)$ son graphe de cohérence et $(i,j) \in \mathcal{S}$.

Si la seconde règle de Condon de la classe *L&P* peut être appliquée sur (i,j),$(i-1,j+1)$ alors pour tout $(l,k) \in \mathcal{S}$ nous avons $cross((i,j),(k,l)) = cross((i+1,j-1),(k,l))$. $(i,j) \in V$ et $(i+1,j-1) \in V$ ont les mêmes voisins dans le graphe de cohérence.

La seule règle permettant d'effacer des arcs croisants est l'application de la règle 3 où $(i,i+2) \in \mathcal{S}$ et $(i+1,i+3) \in \mathcal{S}$ sont supprimés. L'itération de l'application des règles 2 puis de la 3 en dernier lieu nous permet donc de réduire au vide deux ensembles d'arcs empilés qui se croisent mutuellement.

□

Nous devons maintenant prouver qu'il n'y a qu'un seul pseudonœud et que ce pseudonœud n'est pas imbriqué sous un arc.

Lemme 2. *les structures de la classe L&P contiennent au plus un pseudonœud non empilé.*

Démonstration. Seule la troisième règle permet de réduire les arcs croisants. S'il y avait plusieurs structures croisées, il y aurait plusieurs motifs du type *abab* qui ne sauraient être réduits par la règle 3. Donc il n'y a qu'une seule structure de ce type dans la structure.

Si la structure croisée était nichée, alors nous aurions un motif du type *abcbca*. Aucune règle ne permettrait de reduire ni $a \ldots a$ ni *bcbc*. La structure ne peut donc être nichée. □

Ces deux lemmes prouvent le théorème.

2.3 La classe de Dirks et Pierce (*D&P*)

Théorème 5. *Les pseudonœuds dans les structures de la classe D&P sont de type H.*

Démonstration. En utilisant le lemme des structures *L&P* sur la règle 3 de *D&P*, on montre de la même façon que les structures croisées correspondent à des pseudonœuds de type H.

La règle 3 pour *D&P* est applicable pour tout motif de type *abab* dans la structure. Donc l'ensemble d'arcs croisés n'est pas nécessairement unique et peut être niché sous d'autres arcs.

□

Corollaire 1. *Les graphes de cohérence des pseudonœuds de L&P et D&P sont des graphes bipartis complets.*

Démonstration. Immédiat à partir des résultats du théorème précédent.

□

La réciproque est fausse. Par exemple la structure 5.7 à gauche a pour graphe de cohérence $K_{4,2}$ à droite.

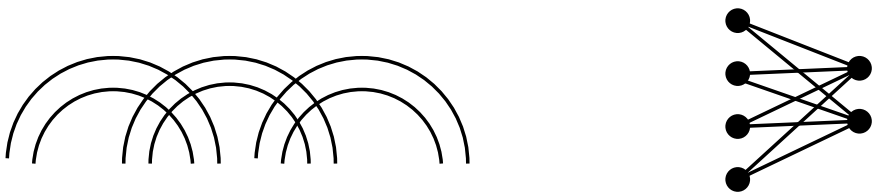


FIGURE 5.7 – ***Structure ARN à gauche et son graphe de cohérence à droite. La structure n'est pas un pseudonœud de type H.***

2.4 Un sous ensemble de la classe de Rivas et Eddy (*R&E*), la classe *G&D* et les figuiers de Barbarie

Avant de décrire la classe *R&E*, nous définissons une classe restreinte des pseudonœuds *R&E*. Cette nouvelle classe possède des propriétés intéressantes et nous permettra une caractérisation plus aisée de la classe *R&E*. Nous appellerons cette classe *G&D* pour Gao et Ding car elle a été décrite par Gao et Ding dans un autre contexte [44]. Cette classe sera plus amplement étudiée dans la section 2.6. Nous ne considérons que les trois premières règles de *R&E* pour caractériser cette classe :

1. $aa \longrightarrow \varepsilon$,
2. $ab \ldots ba \longrightarrow b \ldots b$,
3. $aba \longrightarrow b$.

Définition 39 (Figuiers de Barbarie)**.** *Soit $\mathcal{G} = (V,E)$ un graphe connexe, $\forall u \in V$ on note $N(u)$ l'ensemble des voisins de u. On dit que $\mathcal{G}$ est un figuier de Barbarie si et seulement s'il peut être réduit à un seul sommet par une suite d'opérations parmi les suivantes :*

– *supprimer un sommet de degré 1 ;*
– *supprimer le sommet $u \in V$ s'il existe un sommet v tel que $N(u) = N(v)$.*

Une forêt de figuiers de Barbarie *est un graphe dont chaque composante connexe est un figuier de Barbarie.*

Théorème 6. *Soit $\mathcal{G} = (V,E)$ un graphe. Les deux propositions suivantes sont équivalentes :*
– *$\mathcal{G}$ est le graphe de cohérence d'une structure de la classe* G&D.
– $\mathcal{G}$ est une forêt de figuiers de Barbarie.

La figure 5.8 illustre une structure *G&D* et le figuier de Barbarie correspondant.

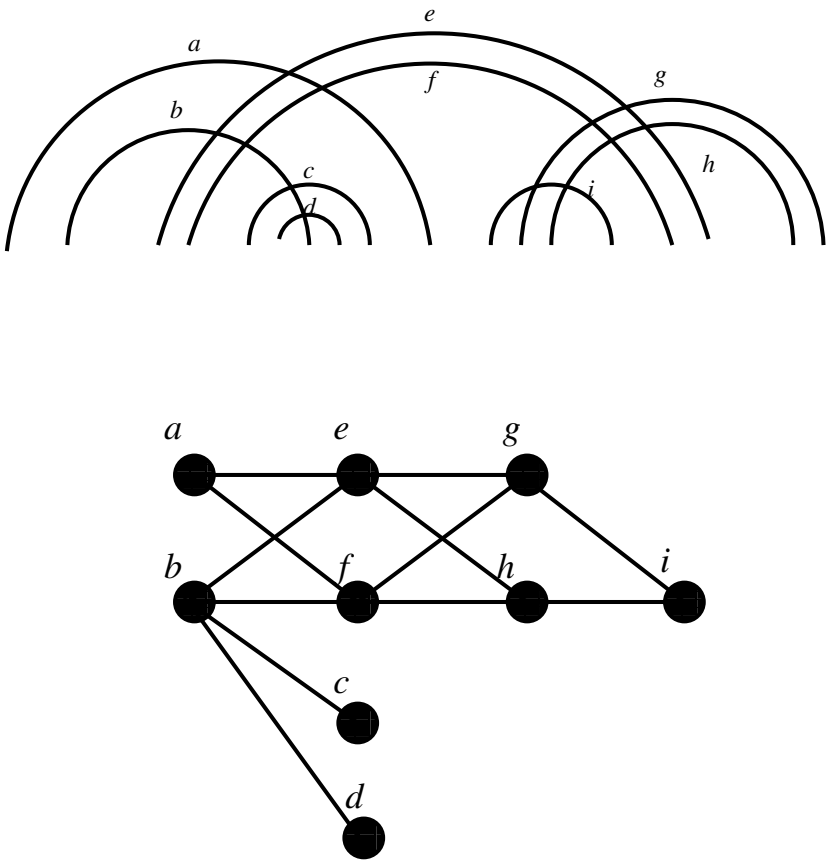


FIGURE 5.8 – ***En haut, une structure G&D. En bas, un figuier de Barbarie correspondant à son graphe de cohérence.***

Pour prouver le théorème, nous montrons d'abord les propriétés suivantes :

Lemme 3. *Si une structure appartient à la classe G&D, son graphe de cohérence est une forêt de figuiers de Barbarie.*

Démonstration. Montrons le par induction sur le nombre d'arcs dans la structure.

Soit $\mathcal{S}$ une structure *G&D* avec $|\mathcal{S}| = n$ le nombre d'arcs et posons $\mathcal{G} = (V,E)$ son graphe de cohérence. Pour $n = 1$, $\mathcal{G}$ est une forêt de figuiers de Barbarie triviale. Supposons maintenant que le graphe de cohérence d'une structure G&D de taille $n-1$ soit une forêt de figuiers de Barbarie et considérons $\mathcal{S}$ de taille n. Comme $\mathcal{S}$

est *G&D*, il existe au moins un arc sur lequel une règle de *G&D* peut être appliquée. Appelons cet arc (i, j). Si la première règle de *G&D* peut être appliquée sur (i, j) nous avons nécessairement $j = i+1$ et (i, j) est un sommet isolé dans $\mathcal{G}$. D'après l'hypothèse d'induction $\mathcal{G}' = \mathcal{G} - \{(i, j)\}$ correspondant à $\mathcal{S}' = \mathcal{S} - \{(i, j)\}$ est une forêt de figuiers de Barbarie. Donc $\mathcal{G}$ est aussi une forêt de figuiers de Barbarie.

Si la troisième règle de *G&D* peut être appliquée sur (i, j) alors nous avons nécessairement $j = i+2$ et le sommet $(i, i+2) \in \mathcal{G}$ est de degré 1. D'après l'hypothèse d'induction, $\mathcal{G}' = \mathcal{G} - \{(i, i+2)\}$ est une forêt de figuiers de Barbarie. L'ajout du sommet $(i, i+2)$ revient à ajouter un sommet de degré 1 dans $\mathcal{G}$ (Figure 5.9). Donc $\mathcal{G}$ est une forêt de figuiers de Barbarie.

Si la seconde règle de *G&D* est applicable sur $(i, j) \in \mathcal{S}$, alors $\forall (l, k) \in \mathcal{S}$ nous avons $cross((i, j), (k, l)) = cross((i+1, j-1), (k, l))$, donc $N_G((i, j)) = N_G((i+1, j-1))$. On dira alors que $\mathcal{G}' = \mathcal{G} - \{(i, j)\}$ est le graphe de cohérence correspondant à la structure $\mathcal{S}' = \mathcal{S} - \{(i, j)\}$ (Figure 5.10). D'après l'hypothèse d'induction, $\mathcal{G}'$ est une forêt de figuiers de Barbarie. L'ajout de (i, j) nous donne une forêt de figuiers de Barbarie.

□

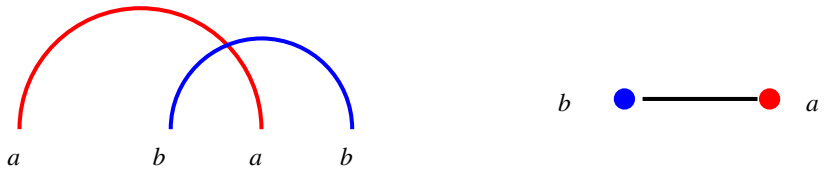


FIGURE 5.9 – ***A gauche, structure illustrant l'application de la règle 3 sur l'arc*** *b****. A droite, le graphe de cohérence correspondant, l'application de la règle 3 provoque la suppression du noeud*** *b****.***

FIGURE 5.10 – ***A gauche, structure illustrant l'application de la règle 2 sur l'arc*** *b****. A droite, le graphe de cohérence correspondant, l'application de la règle 2 provoque la suppression du noeud*** *b****.***

Pour montrer la propriété réciproque, nous devons introduire la notion de mouvement conservatif.

Définition 40 (Mouvement conservatif)**.** *Soient $\mathcal{S}$ une structure et $\mathcal{G} = (V, E)$ le graphe de cohérence correspondant. Un mouvement conservatif est une transformation de $\mathcal{S}$ vers $\mathcal{S}' \neq \mathcal{S}$ tel que $\mathcal{G} = \mathcal{G}'$*

Lemme 4. *Si $\mathcal{S}$ est dans G&D et son graphe de cohérence est $\mathcal{G}$, toutes les structures $\mathcal{S}'$ dont le graphe de cohérence $\mathcal{G}'$ est isomorphe à $\mathcal{G}$ sont dans G&D.*

Démonstration. Nous allons construire une instance pour chacune des règles telle que seule la règle considérée puisse être appliquée. Nous allons aussi décrire le contexte de ces arcs. Nous allons construire les arcs a et b impliqués dans la règle considérée, puis les arcs empilés sur a et b et ceux imbriqués dessous. Afin que seuls les arcs a et b soient impliqués dans une règle on construit des arcs qui empêchent l'application des trois règles de la classe G&D. Les arcs ne croisant pas directement les arcs a et b ne nous intéressent pas puisqu'ils n'interviennent pas localement sur a et b. De même, des configurations plus complexes ne nous apporteraient rien de plus puisque les règles de Condon sont des règles locales, un seul arc nous suffit donc. Construisons cette structure pour les règles de ré-écriture 2 et 3 de *G&D* (La règle 1 est triviale). Nous obtenons la structure présentée dans la figure 5.11 pour la règle 2. Nous obtenons la structure présentée dans la figure 5.12 pour la règle 3. Plusieurs structures similaires peuvent être obtenues à des symétries près. On vérifie facilement que ces structures sont de la classe *G&D*.

□

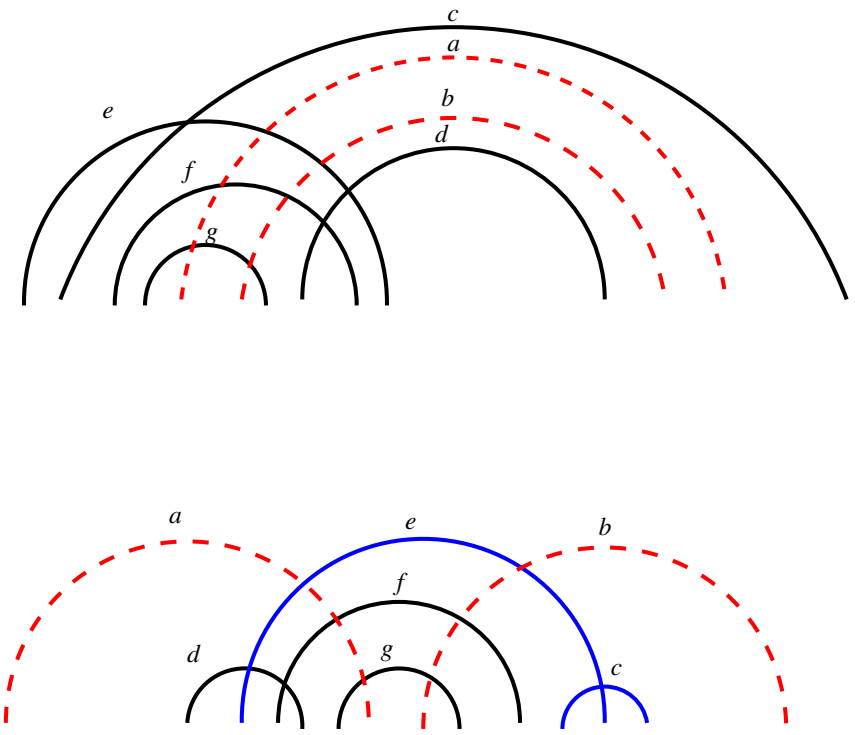


FIGURE 5.11 – ***En haut, seule la règle 2 est applicable sur les arcs en pointillés rouges. En bas, après mouvement conservatif, la règle 3 est applicable sur les arcs pleins bleus.***

Corollaire 2. *Si un graphe de cohérence est un figuier de Barbarie alors sa structure correspondante est dans la classe G&D.*

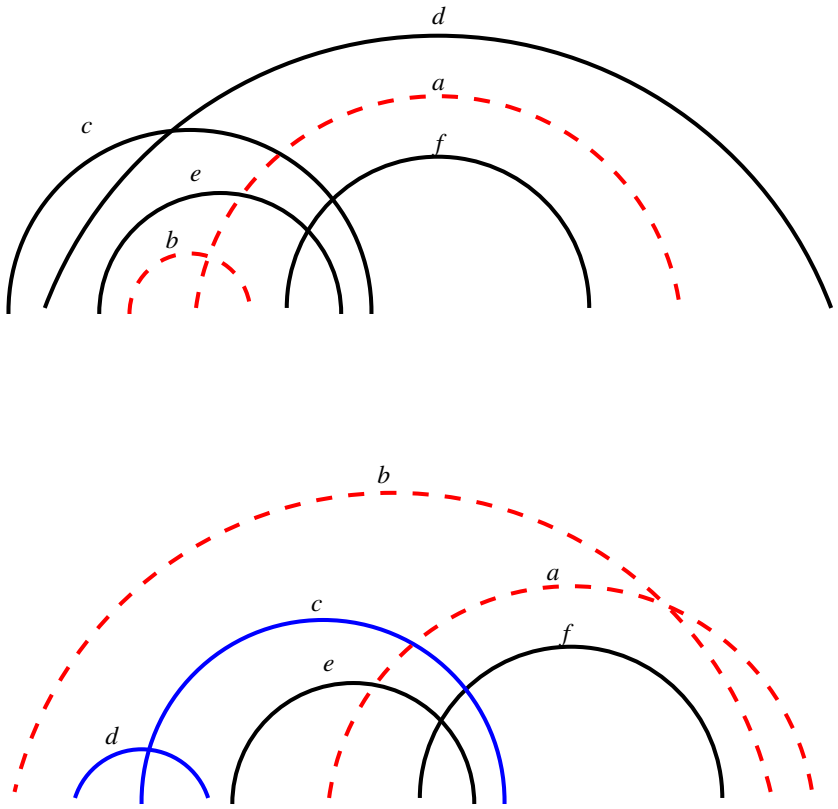


FIGURE 5.12 – ***En haut, seule la règle 3 est applicable sur les arcs en pointillés rouges. En bas, après mouvement conservatif, la règle 3 est applicable sur les arcs pleins bleus.***

2.5 La classe de Rivas et Eddy (*R&E*)

Définition 41 (Figuier de Barbarie à cliques)**.** *Les figuiers de Barbarie à cliques peuvent être réduits à un sommet grâce à une suite des opérations suivantes :*

- *supprimer les sommets de degré 1 ;*
- *supprimer les sommets $u \in V$ s'il existe un sommet v tel que $N(u) = N(v)$*
- *supprimer les sommets $u \in V$ s'il existe un sommet v tel que $N(u) \cup \{u\} = N(v) \cup \{v\}$*

Théorème 7. *Les propositions suivantes sont équivalentes :*

- *Le graphe $\mathcal{G}$ est un graphe de cohérence pour les structures de R&E*
- *Le graphe $\mathcal{G}$ est un figuier de Barbarie à cliques.*

Démonstration. Montrons cette propriété par induction sur le nombre d'arcs dans la structure.

$\mathcal{S}$ est une structure de la classe *R&E* avec $|\mathcal{S}| = n$ le nombre d'arcs, $\mathcal{G} = (V, E)$ est son graphe de cohérence.

La propriété est vraie pour $n = 1$.

Considérons la propriété vraie pour $n-1$. Nous devons la vérifier pour n. La propriété a été vérifié pour les trois premières règles de *G&D*. Nous ne devons donc vérifier que la dernière règle.

Si la règle 4 de la classe *R&E* peut être appliquée sur le couple $(i,j),(i+1,j+1) \in \mathcal{S}$, alors $\forall (l,k) \in \mathcal{S}$ nous avons $cross((i,j),(k,l)) = cross((i+1,j+1),(k,l))$ and $cross((i,j),(i+1,j+1)) = vrai$. On a donc $N(u) \cup \{u\} = N(v) \cup \{v\}$. Le graphe $\mathcal{G}' = \mathcal{G} - \{(i,j\})$ correspond à la structure $\mathcal{S}' = \mathcal{S} - (i,j)$ (Figure 5.13). Par hypothèse d'induction le graphe $\mathcal{G}'$ est un figuier de Barbarie à clique, donc $\mathcal{G}$ en est un aussi.

□

FIGURE 5.13 – ***A gauche, structure illustrant l'application de la règle 4 sur l'arc b. A droite, le graphe de cohérence correspondant, l'application de la règle 4 provoque la suppression du noeud b..***

Lemme 5. *Les structures $\mathcal{S}$ de la classe R&E sont stables par mouvement conservatif.*

Démonstration. Similaire à la preuve de *G&D*. Nous montrons qu'un mouvement conservatif de la règle 4 de *R&E* maintient la classe. La figure 5.14 illustre ce mouvement.

□

2.6 Résultat alternatif de Gao et Ding (*G&D*)

Dans leur papier de 2008, Gao et Ding [44] se sont intéressés à la restriction des structures de la classe *R&E* aux structures bisecondaires. Les structures bisecondaires ont été précédemment décrite par Hasslinger et Stadler [53] et nommées pour cette raison *H&S* par Rødland [92]. Gao et Ding montrent que $G\&D = H\&S \cap R\&E$.

Nous donnons ici une preuve alternative en utilisant les propriétés des graphes de cohérence.

Lemme 6. *Si $\mathcal{S} \in R\&E, \mathcal{S} \notin G\&D$ et $\mathcal{G}$ est son graphe de cohérence, alors $\mathcal{G}$ contient K_3.*

Démonstration. Montrons le récursivement sur $|\mathcal{S}| = n$.

La plus petite structure telle que $\mathcal{S} \in R\&E, \mathcal{S} \notin G\&D$ qui vérifie la propriété est $(1,4),(2,5),(3,6)$. Supposons la propriété vraie à l'ordre $n-1$ et vérifions la à l'ordre n.

Si $\mathcal{S} \in R\&E$ et $\mathcal{S} \notin G\&D$, cela signifie que $\mathcal{S}$ ne peut être ré-écrite sans utiliser la règle 4.

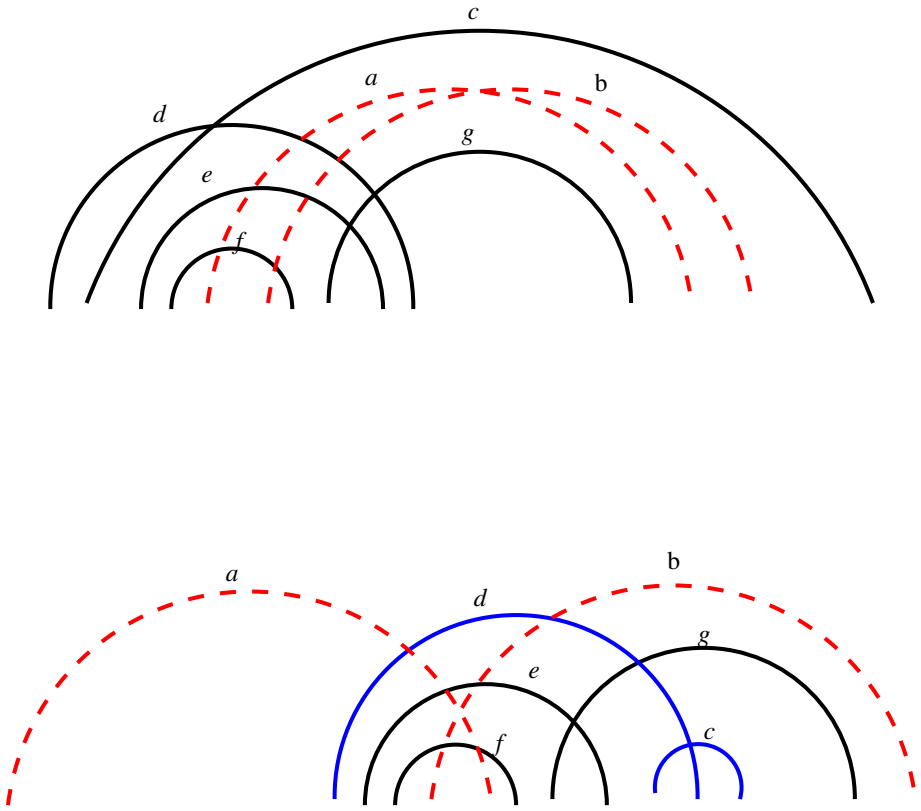


FIGURE 5.14 – ***En haut, seule la règle 4 est applicable sur les arcs en pointillés rouges. En bas, après mouvement conservatif, la règle 3 est applicable sur les arcs pleins bleus.***

Si $\nexists (a,b) \in \mathcal{S}, 1 \leq a < i, i+1 < b < j$ ou $i+1 < a < j, j+1 < b \geq n$ alors $\mathcal{S}$ peut être divisée en deux sous-structures R&E de taille inférieure à n. Par hypothèse d'induction, ces structures sont R&E et leur graphe de cohérence contient K_3.

Sinon, nous avons $cross((i,j),(a,b)) = cross((i+1,j+1),(a,b)) = vrai$ comme $cross((i,j),(i+1,j+1)) = vrai$ nous en déduisons que le graphe de cohérence $\mathcal{G}$ de $\mathcal{S}$ contient K_3

□

En conséquence :

Théorème 8. *G&D = R&E ∩ H&S (Figure 5.15)*

Lemme 7. *G&D ⊂ H&S*

Démonstration. D'après [53], les graphes de cohérence des structures de la classe *H&S* sont bipartis. Nous allons montrer que les graphes *G&D* le sont aussi.

Soit $\mathcal{G} = (V,E)$ un figuier de Barbarie.

La première règle de ré-écriture des figuiers de Barbarie autorise la suppression des sommets de degré 1. Cette opération autorise seulement la suppression de sous-graphes bipartis.

Soit $u \in V$ et $v \in V$ et $N_G(u) = N_G(v)$. Si $|N_G(v)| = 1$, u et v sont de degré 1 et la règle précédente s'applique. Sinon considérons $w \in N_G(u)$ et $w' \in N_G(u)$ alors $uwvw'$ forme un cycle de taille 4.

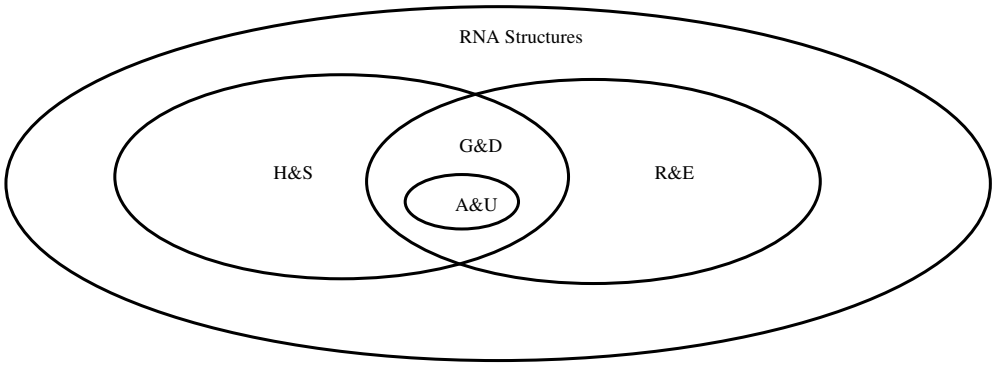


FIGURE 5.15 – ***Classification de Condon** et al **enrichie de la classe** G&D.*

Donc la seconde opération de réduction des figuiers de Barbarie implique seulement la suppression de sommets impliqués dans des cycles de taille 4. Cette opération permet donc de réduire des sous-graphes bipartis.

Les deux opérations ne peuvent supprimer que des sommets impliqués dans des sous-graphes bipartis. Donc les graphes de cohérence de la classe G&D sont bipartis.

□

Preuve du théorème 8. Par construction, nous savons que $G\&D \subset R\&E$ et nous avons montré que $G\&D \subset H\&S$. Le lemme précédent a montré que K_3 est un sous-graphe des graphes de cohérence de la classe $R\&E$ mais pas de ceux de la classe $G\&D$, donc $(R\&E \setminus G\&D) \cap H\&S = \emptyset$. Les graphes $C_{2n}, n \geq 3$ (Figure 5.16) sont des graphes de cohérence de la classe H&S mais pas des graphes de cohérence de $G\&D$. Donc $G\&D \neq H\&S$.

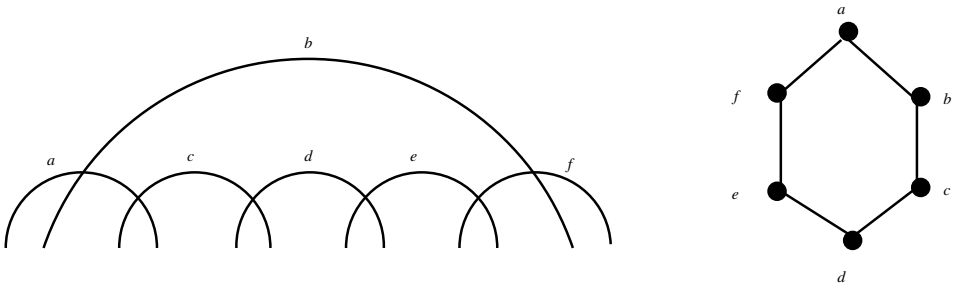


FIGURE 5.16 – ***A gauche, structure*** H&S et son graphe de cohérence C_6 à droite.

On en déduit $G\&D = R\&E \cap H\&S$.

□

Théorème 9. $A\&U \subset G\&D$

Démonstration. La troisième règle de $A\&U$ est une restriction de la troisième règle de $G\&D$

□

Chapitre 6

Dénombrement et génération aléatoire de structures avec pseudonœuds

Dans cette section nous allons étudier le compromis entre complexité en temps et espace de prédiction des algorithmes exacts *ab* initio de structures d'ARN. Pour cela, nous allons montrer que l'on peut décrire les pseudonœuds simples et les structures simplement pseudonouées grace à des langages non contextuels. Nous pourrons alors utiliser les outils décrits dans les sections 1 et 4 pour déduire les formules asymptotiques de dénombrement des structures prédites par les différents algorithmes (section 1). Ce travail fera prochainement l'objet d'une publication dans Journal of Computational Biology [95]. Dans la section 2, nous donnerons deux bijections entre certaines classes de pseudonœuds et des objets combinatoires connus afin d'obtenir les formules closes d'énumération. Enfin, nous appliquerons ces résultats à la génération aléatoire de structures avec pseudonœuds dans la section 3.2. Ce dernier travail a fait l'objet d'une publication dans la conférence de génération aléatoire GASCOM [96].

1 Etude du compromis entre complexité en temps et espace de prédiction des algorithmes

1.1 Un modèle algébrique pour les structures avec pseudonœuds

Nous allons voir comment nous pouvons coder les pseudonœuds. Comme nous l'avons vu, toutes les classes sauf $A\&U$ contiennent des pseudonœuds de type H. Nous verrons que les pseudonœuds de la classe $A\&U$ sont simples. Les pseudonœuds de la classe $R\&E$ et sa restriction planaire $G\&D$ ne seront pas étudiés. Ici, nous définissons une transformation qui autorise le codage de n'importe quelle classe de structures pseudonouée pour lesquelles tous les pseudonœuds sont simples, grâce à un langage non sensible au contexte. Rappelons quelques définitions.

Définition 42. *Soit L un langage défini sur un alphabet A, et $w = w_1 w_2 \dots w_n$ un mot de L, où w_i sont les lettres de w. Un mot v est un* sous-mot *de w si $v = w_{i_1} w_{i_2} \dots w_{i_k}$,*

où $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. *La* projection *de w sur un alphabet* $A' \subset A$ *est un sous-mot* w' *obtenu en effaçant de w toutes les lettres qui n'appartiennent pas à* A'. *La projection de L sur* A' *est l'ensemble des projections des mots de L sur* A'.

Enfin, rappelons que le langage de Dyck sur un alphabet à deux lettres $\{d, \bar{d}\}$ est le langage des mots bien parenthésés, où d et $\bar{d}$ représentent respectivement les parenthèses ouvrante et fermante.

Nous pouvons dès lors établir naturellement les deux lemmes qui suivent :

Lemme 8. *Toute classe de structures pseudonouées où tous les pseudonœuds sont simples peuvent être codés par les mots du langage L sur l'alphabet* $\{d, \bar{d}, x, \bar{x}, y, \bar{y}, c\}$ *où*

- d *et* $\bar{d}$ *codent, respectivement, les extrémités gauches et droites des arcs qui ne sont pas impliquées dans un pseudonœud ;*
- x *et* $\bar{x}$ *codent, respectivement, les extrémités gauches et droites des arcs qui sont impliquées dans la partie gauche d'un pseudonœud ;*
- y *et* $\bar{y}$ *codent, respectivement, les extrémités gauches et droites des arcs qui sont impliquées dans la partie droite d'un pseudonœud ;*

De plus, la projection du langage sur l'alphabet $\{d, \bar{d}\}$ *(resp.* $\{x, \bar{x}\}$, $\{y, \bar{y}\}$*) est un sous langage de Dyck sur le même alphabet.*

Lemme 9. *Soit S une structure simplement pseudonouée, et w le mot sur* $\{d, \bar{d}, x, \bar{x}, y, \bar{y}\}$ *qui code S. Alors, chaque pseudonœud simple de S est codé par un sous-mot v de w, tel que*

$$v = x^n y^{m_1} \bar{x}^{n_1} y^{m_2} \bar{x}^{n_2} \ldots y^{m_k} \bar{x}^{n_k} \bar{y}^m$$

où $n_1 + n_2 + \ldots + n_k = n$ *et* $m_1 + m_2 + \ldots + m_k = m$.

On remarque qu'un pseudonœud de type H est un pseudonœud simple où $k = 1$. Dès lors chaque pseudonœud de type H dans S est codé par un sous-mot
$v = x^n y^m \bar{x}^n \bar{y}^m$.

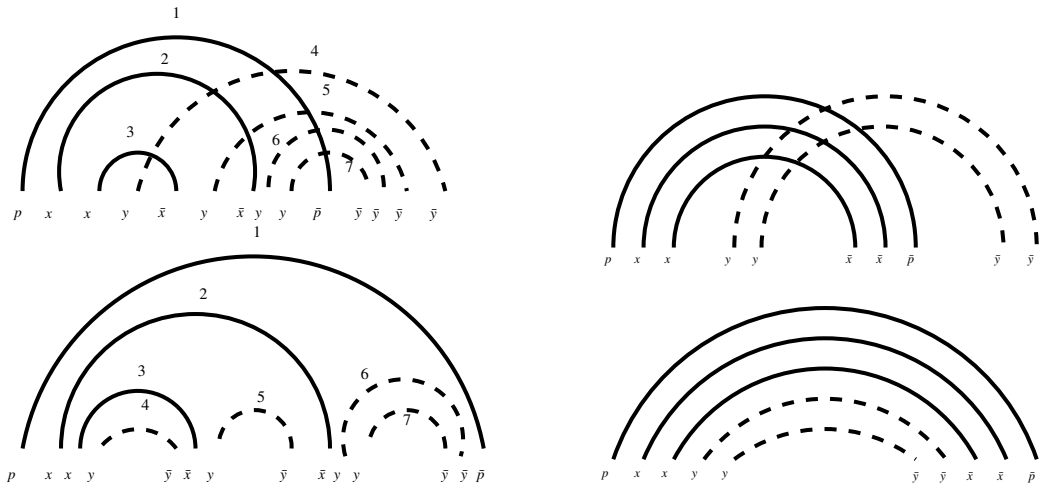
Enfin, la proposition suivante donne une façon de coder toute structure pseudonouée où tous les pseudonœuds sont simples par un sous-ensemble du langage de Dyck avec quatre sortes de parenthèses, c'est à dire sur l'alphabet $\{d, \bar{d}, x, \bar{x}, y, \bar{y}, p, \bar{p}\}$.

Proposition 2. *Soit S comme une structure pseudonouée, et w comme un mot sur* $\{d, \bar{d}, x, \bar{x}, y, \bar{y}\}$ *qui code S. Alors w peut être codé par un mot où chaque sous-mot* $v = x^n y^{m_1} \bar{x}^{n_1} y^{m_2} \bar{x}^{n_2} \ldots y^{m_k} \bar{x}^{n_k} \bar{y}^m$ *correspondant à un pseudonœud de type H est remplacé par :* $v' = p\, x^{n-1} y^{m_1} \bar{y}^{m_1} \bar{x}^{n_1} y^{m_2} \bar{y}^{m_2} \bar{x}^{n_2} \ldots y^{m_k} \bar{y}^{m_k} \bar{x}^{n_k - 1} \bar{p}$.

En particulier, chaque sous-mot $v = x^n y^m \bar{x}^n \bar{y}^m$ correspondant à un pseudonœud simple est remplacé par $v' = p\, x^{n-1} y^m \bar{y}^m \bar{x}^{n-1} \bar{p}$.

Démonstration. Nous avons une correspondance immédiate entre les deux sortes de mots. La transformation est illustrée dans la figure 6.1(a) et la figure 6.1(b), respectivement, pour les pseudonœuds simples et pour le cas particulier des pseudonœuds de type H.

□



(a) Pseudonœud simple (b) Pseudonœud de type H

FIGURE 6.1 – ***En haut : Le pseudonœud et son codage v. les structures empilées et son codage v' donnée par la proposition 2. Les lignes pleines représentent x et $\bar{x}$ et p et $\bar{p}$, les lignes pointillées représentent y et $\bar{y}$.***

1.2 Dénombrement asymptotique des classes

A partir des modèles de croisement décrits en 1.1, on peut écrire les grammaires décrivant les différentes classes de pseudonoeuds. Ces grammaires nous permettent d'obtenir les comportements asymptotiques des séries génératrices énumérant les mots décrits par ces grammaires. Nous allons donner les grammaires correspondant aux structures décrites par la classification de Condon [25]. Nous ajouterons les deux classes de Cao et Chen [18] et de Reeder et Giegerich [87].

Pour chacune des classes *D&P*, *A&U*, *R&G* et *C&C*, nous allons donner un équivalent asymptotique du nombre de structures prédictibles de taille n. Dans chaque cas, la preuve se déroule en trois étapes :

1. Nous construisons une grammaire non contextuelle qui engendre le langage qui code les structures considérées. La grammaire est bâtie à partir de la grammaire généraliste donnée en 2.2 et de le codage donné dans la proposition 2.
2. A partir de la grammaire, nous pouvons déduire une équation algébrique qui satisfait la fonction génératrice ordinaire (o.g.f.) du langage. On construit cette équation selon la méthodologie DSV décrite en section 1.
3. A partir de cette équation nous calculons une formule asymptotique pour le nombre de structures de taille n. Nous utilisons la méthode décrite dans le chapitre 2.

Pour chaque classe $X\&Y$, nous écrivons $X\&Y(n)$ pour le nombre de structures ayant n arcs.

La classe $A\&U$.

D'après [2, 25], les structures A&U sont composées d'arcs non croisés, les pseudonœuds sont simples.

Théorème 10.

$$A\&U(n) \;=\; \frac{\alpha_1}{2\sqrt{\pi}}\omega_1{}^n n^{-3/2}(1+O(1/n)),$$

où $\alpha_1 = 0.6575407644...,\ \omega_1 = 7.547308334...$, *sont des constantes algébriques.*

Démonstration. Soit $L_{A\&U}$ le langage qui code la classe $A\&U$ en accord avec le codage de la proposition 2. La grammaire non contextuelle et non ambiguë suivante engendre $L_{A\&U}$:

$$\begin{aligned} S &\rightarrow dS\bar{d}S|P \\ P &\rightarrow pSX\bar{p}S|\varepsilon \\ X &\rightarrow xSX\bar{x}SY|yYS\bar{y}S \\ Y &\rightarrow ySY\bar{y}S|\varepsilon \end{aligned}$$

Les deux règles de la première ligne permettent d'engendrer les arcs non croisants en position quelconque. Le autres règles engendrent des mots qui correspondent au codage des pseudonœuds simples comme le montre la figure 6.2.

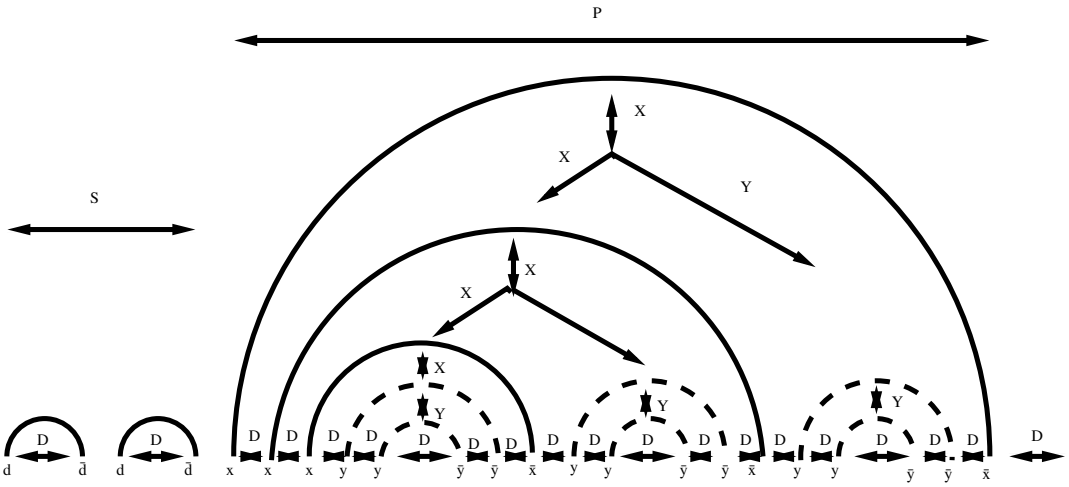


FIGURE 6.2 – ***Construction d'une structure selon la grammaire de*** $L_{A\&U}$**.**

Etant donné la grammaire, nous obtenons le système d'équations pour l'o.g.f. Posons le symbole formel z qui dénote un arc, nous obtenons directement la traduction suivante :

$$\begin{aligned} S(z) &= zS^2(z)+P(z) \\ P(z) &= zS^2(z)X(z)+1 \\ X(z) &= zS^2(z)Y(z)(X(z)+1) \\ Y(z) &= zS^2(z)Y(z)+1. \end{aligned}$$

On obtient par substitution l'o.g.f. $S(z)$ est la solution de l'équation algébrique :

$$F(z,S) \;=\; z^2S^4 - 2zS^3 + zS^2 + S - 1 = 0, \tag{6.1}$$

à partir de laquelle nous déduisons le nombre de structures de taille n.

La détermination de la formule asymptotique se déroule conformément aux indications données en 2. Nous calculons ainsi $\sigma_1 = 1.403556586...$ et $\rho_1 = 0.1324975681...$ à partir desquels on calcule α_1 et ω_1. □

La classe *D&P*.

Les structures de la classe *D&P* sont caractérisées par la présence d'arcs non croisés et de pseudonœuds de type H en nombre quelconque [33, 25].

Théorème 11.

$$D\&P(n) \;=\; \frac{\alpha_2}{2\sqrt{\pi}}\omega_2{}^n n^{-3/2}(1+O(1/n)),$$

où $\alpha_2 = 0.7534777262...$, $\omega_2 = 7.3148684640...$, *sont des constantes algébriques.*

Démonstration. La grammaire suivante non ambiguë engendre le langage qui code les structures D&P d'après la proposition 2 :

$$\begin{aligned} S &\rightarrow dS\bar{d}S|P \\ P &\rightarrow pXS\bar{p}S|\varepsilon \\ X &\rightarrow xSX\bar{x}S|ySY\bar{y}S \\ Y &\rightarrow ySY\bar{y}S|\varepsilon \end{aligned}$$

La première ligne permet d'engendrer des structures sans pseudonœud et de les placer partout dans la séquence par l'intermédiaire du non terminal P. Les trois dernières lignes engendrent les mots qui correspondent au code pour les pseudonœuds de type H. P engendre le premier arc de l'ensemble d'arcs gauche. Les autres arcs de cet ensemble sont engendrés par l'intermédiaire de X. Le symbole Y engendre les arcs dans la partie droite.

A partir de cette grammaire on déduit l'équation suivante :

$$F(z,S) = z^3S^6 - z^2S^5 + 2zS^3 - zS^2 - S + 1 = 0 \tag{6.2}$$

Cette équation est similaire à celle obtenues pour *A&U* et est résolue de la même manière. On trouve la singularité dominante $z = \rho_2 = 0.1367078581...$, $S = \sigma_2 = 1.439796009...$, et on en déduit $\alpha_2 = 0.7534777262...$, $\omega_2 = 7.3148684640...$,. Il est notable dans ce système que $z = \mu = 0.08794976637...$, $S = \tau = 7.169944393...$, est une autre singularité de module moins élevé que la première. Cependant cette singularité ne se trouve pas sur le prolongement de la courbe qui part de l'origine. Elle ne peut donc constituer la singularité dominante de $S(z)$ et on ne la prend donc pas en considération.

□

la classe *R&G*.

Théorème 12.

$$R\&G(n) \;=\; \frac{\alpha_3}{2\sqrt{\pi}}\omega_3{}^n n^{-3/2}(1+O(1/n)),$$

où $\alpha_3 = 1.165192913...,\ \omega_3 = 6.576040092...$, *sont des constantes algébriques.*

Démonstration. Dans [87], on donne la grammaire suivante (à laquelle nous avons enlevé les bases non appariées) :

$$S \to SS|dS\bar{d}|x^k Sy^l S\bar{x}^k S\bar{y}^l|\varepsilon.$$

Cette grammaire n'est pas non contextuelle. Cependant, nous remarquons que les pseudonœuds décrits sont des cas particuliers de pseudonœuds de type H. On applique du fait le codage de la proposition 2 à nouveau et nous obtenons la grammaire non contextuelle suivante :

$$\begin{aligned} S &\to dS\bar{d}S|P \\ P &\to pX\bar{p}S|\varepsilon \\ X &\to xX\bar{x}|SyY\bar{y}S \\ Y &\to yY\bar{y}|S \end{aligned}$$

dont on déduit l'équation algébrique :

$$F(z,S) = z^2S^4 + z(z-1)^2S^2 - (z-1)^2S + (z-1)^2 = 0 \tag{6.3}$$

Cette équation est similaire aux précédentes. Nous la résolvons de la même manière et trouvons la singularité dominante $z = \rho_3 = 0.1520671994...,\ S = \sigma_3 = 1.589450164...$ dont on déduit $\alpha_3 = 1.165192913...,\ \omega_3 = 6.576040092...,$

□

La grammaire donnée dans la preuve du théorème 12 nous permet de positionner cette nouvelle classe dans la classification de Condon *et al.*

Théorème 13. *$R\&G \subset D\&P$, $L\&P \cap R\&G \neq \emptyset$ and $R\&G \not\subset L\&P$*

Démonstration. La grammaire qui décrit les pseudonœuds de *R&G* est moins générale que la grammaire pour les pseudonœuds de type H. On a donc $R\&G \subset D\&P$ et $L\&P \cap R\&G \neq \emptyset$. Comme les structures R&G peuvent contenir plusieurs pseudonœuds, on a $L\&P \cap R\&G \neq L\&P$.

□

La classe *C&C*.

Théorème 14.

$$C\&C(n) \;=\; \frac{\alpha_4}{2\sqrt{\pi}}\omega_4{}^n n^{-3/2}(1+O(1/n)),$$

où $\alpha_4 = 1.665071176...,\ \omega_4 = 5.856765093...$, *sont des constantes algébriques.*

Démonstration. La grammaire non contextuelle suivante engendre les structures *C&C* :

$$S \to SS|dSd\bar{l}|x^kSy^l\bar{x}^kS\bar{y}^l|\varepsilon.$$

Elle peut être transformée en grammaire non contextuelle qui est une restriction de la grammaire *R&G*.

$$\begin{aligned} S &\to dS\bar{d}S|P \\ P &\to pX\bar{p}S|\varepsilon \\ X &\to xX\bar{x}|SyY\bar{y}S \\ Y &\to yY\bar{y}|\varepsilon \end{aligned}$$

On obtient alors pour la f.g.o. des structures *C&C* :

$$F(z,S) = z^2S^3 + z(z-1)^2S^2 - (z-1)^2S + (z-1)^2 = 0. \tag{6.4}$$

Cette équation est similaire aux précédentes. On trouve la singularité $z = \rho_4 = 0.1707427197...$, $S = \sigma_4 = 1.7663614360...$, et on en déduit $\alpha_4 = 1.665071176...$, $\omega_4 = 5.856765093...$, □

De plus, on montre aisément que :

Théorème 15. *C&C* $\subset$ *D&P, L&P* $\cap$ *C&C* $\neq 0$, *C&C* $\not\subset$ *L&P et C&C* $\subset$ *R&G*

La classe *L&P*.

Nous avons déjà donné une formule close et un équivalent asymptotique pour cette classe dans la section 2.1. Nous montrons le théorème 17 d'une autre façon.

Nous montrons que toute structure *L&P* peut être codée par un mot issu d'un langage non contextuel non ambiguë. Les calculs habituels nous conduisent à l'obtention de la fonction génératrice.

Théorème 16.

$$L\&P(n) \;=\; \frac{1}{2}\,4^n(1+O(n^{-1/2})).$$

Démonstration. Toute structure de la classe *L&P* de taille *n* peut être codée par un mot de taille *n* issu d'un langage non sensible au contexte décrit par la grammaire non ambiguë suivante :

$$\begin{aligned} S &\to dD\bar{d}S|P \\ D &\to dD\bar{d}D|\varepsilon \\ P &\to xDX\bar{x}D|\varepsilon \\ X &\to xDX\bar{x}D|yDY\bar{y}D \\ Y &\to yDY\bar{y}D|\varepsilon \end{aligned}$$

Le système d'équations dont la f.g.o. $S(z) = \Sigma_n L\&P(n)z^n$ est solution (n est le nombre d'arcs) se déduit du système suivant :

$$\begin{aligned} S(z) &= zS(z)D(z) + P(z) \\ D(z) &= zD(z)^2(z) + 1 \\ P(z) &= zD^2(z)X(z) + 1 \\ X(z) &= zD^2(z)(X(z) + Y(z)) \\ Y(z) &= zY(z)D^2(z) + 1 \end{aligned}$$

La série $D(z)$ est aisément identifiée à celle comptant les mots du langage de Dyck : $D(z) = (1 - \sqrt{1-4z})/2z$. Contrairement aux cas précédents, ce système peut être résolu explicitement puisque les autres équations sont linéaires et que le système est clairement trigonal. On obtient succesivement $Y(z)$, $X(z)$, $P(z)$ et $S(z)$ en utilisant le fait que $zD^2(z) = D(z) - 1$. A la fin on trouve :

$$S(z) = \frac{8(1 - 5z + 5z^2 + (3z-1)\sqrt{1-4z})}{(1-4z-\sqrt{1-4z})^2(1+\sqrt{1-4z})}.$$

Le dénominateur s'annule pour $z = 0$ et $z = \frac{1}{4}$, mais $S(z)$ n'est pas singulier à l'origine. On a donc le développement de Taylor :

$$S(z) = 1 + z + 3z^2 + 12z^3 + 51z^4 + 218z^5 + 926z^6 + 3902z^7 + O(z^8)$$

La singularité dominante est $z = \rho_5 = 1/4$ et $S(z)$ admet le développement en $\sqrt{1-4z}$ suivant :

$$S(z) \;=\; \frac{1}{2}\frac{1}{1-4z} - \frac{3}{2}\frac{1}{\sqrt{1-4z}} + 8 - 4\sqrt{1-4z} + O(1-4z).$$

En conséquence, les coefficients de $S(z)$ ont le développement asymptotique suivant :

$$L\&P(n) = \frac{1}{2}\,4^n - \frac{3}{2\sqrt{\pi}}\,4^n n^{-1/2} + \frac{4}{2\sqrt{\pi}} 4^n n^{-3/2}(1 + O(1/n)).$$

□

Nous avons prouvé que chaque classe de pseudonoeuds qui peut être prédite par un algorithme exact (à l'exception de *R&E* pour lequel le problème reste ouvert) peut être codée par un langage algébrique. Nous avons étendu la classification de Condon *et al.* par l'ajout de deux classes et nous avons les formules closes ou asymptotiques pour la cardinalité de toutes les classes à l'exception de *R&E*.

Les résultats pour le dénombrement, résumés dans la Table 1.2, nous permettent de quantifier la relation entre la complexité d'un algorithme et le degré de généralité

Classes	asymptotique	α	ω	Complexité	Remarques
PKF	$\frac{\alpha}{2\sqrt{\pi}n^{3/2}}\omega^n$	2	4	$O(n^3)$	Nombres de Catalan
L&P	$\frac{1}{2}\omega^n$	-	4	$O(n^5)$	Formule close
C&C	$\frac{\alpha}{2\sqrt{\pi}n^{3/2}}\omega^n$	1,6651	5,857	$O(n^6)$	
R&G	$\frac{\alpha}{2\sqrt{\pi}n^{3/2}}\omega^n$	0,1651	6,576	$O(n^4)$	
D&P	$\frac{\alpha}{2\sqrt{\pi}n^{3/2}}\omega^n$	0,7535	7,315	$O(n^5)$	
A&U	$\frac{\alpha}{2\sqrt{\pi}n^{3/2}}\omega^n$	0,6575	7,547	$O(n^5)$	
R&E	Ouvert	-	-	$O(n^6)$	
Tous	$\sqrt{2}\cdot 2^n\cdot\left(\frac{n}{e}\right)^n$	-	-	NPC	Involutions sans point fixe

TABLE 6.1 – Tableau récapitulatif des formules de dénombrement asymptotiques des différentes classes avec la complexité en temps des algorithmes.

de la classe correspondante. Notemment, d'un point de vue strictement quantitatif, l'augmentation de complexité d'un facteur de n^2 entre les classes *PKF* et *L&P* ne semble pas être justifiée comparée au faible gain de cardinalité. Ce fait se comprend si l'on considère qu'une structure de la classe *L&P* n'est qu'une structure sans pseudonoeud à laquelle on ajoute un pseudonoeud de type H. La situation est même pire pour la classe *C&C*, dont la complexité de l'algorithme est plus importante que celle de l'algorithme de *R&G* tandis que $C\&C \subset R\&G$ et le ratio de leur cardinalité est exponentiel. Il faut nuancer ce résultat par le fait que l'algorithme de *C&C* calcule la fonction de partition des structures considérées selon un modèle thermodynamique très élaboré. La croissance linéaire de complexité entre l'algorithme de *R&G* et celui de *PKF* semble très raisonnable comparée à la croissance exponentielle de la cardinalité du nombre de structures. *A&U* et *D&P* ont la même complexité tandis que la croissance de cardinalité est exponentielle en passant de *D&P* à *A&U*. Cependant *D&P* calcule la fonction de partition et *A&U*, bien qu'exponentiellement plus grande ne prédit pas plus de structures biologiquement significatives [85].

2 Résultats bijectifs

Dans cette section, nous allons présenter deux nouvelles bijections qui vont nous permettre d'obtenir les formules closes de dénombrement de la classe *L&P* ainsi que d'une classe des pseudonœuds indifférenciés.

2.1 La classe de *L&P* et cartes planaires

La classe de Lyngsø-Pedersen (L&P) est la plus simple des classes de structures avec pseudonœud. Selon [71] et [25], une structure est dans L&P si et seulement si elle contient au plus un pseudonœud de type H. Ce pseudonœud ne doit pas être situé sous un autre arête. On peut trouver des structures empilées entre deux extrémités d'un arête du pseudonœud. Le théorème 17 ainsi que le corollaire 3 qui en découle, donnent respectivement les formules closes et asymptotiques du nombre

de ces structures.

Théorème 17. *Le nombre de structures* L&P avec n arêtes est :

$$LP(n) = \frac{1}{2}\cdot 4^n - \binom{2n+1}{n} + \binom{2n-1}{n-1} + \frac{1}{n+1}\binom{2n}{n}.$$

Corollaire 3.

$$LP(n) \sim \frac{1}{2}\cdot 4^n.$$

Démonstration. La preuve est bijective : nous établissons une bijection entre l'ensemble des structures L&P de taille n et l'ensemble des cartes planaires enracinées sans isthme à un ou deux sommets ayant n arêtes. les trois premiers termes calculent les cartes à deux sommets [97, 109], tandis que le dernier terme, le nombre de Catalan, compte le nombre de cartes à un sommet. [105].

Une *carte planaire* est le plongement d'un graphe planaire dans le plan.

On dit qu'elle est *sans isthme* si la suppression de n'importe quelle arête ne déconnecte pas le graphe (Figure 6.3).

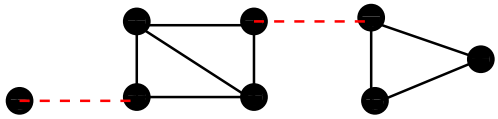


FIGURE 6.3 – ***Exemple de carte planaire avec deux isthmes (en pointillés rouges).***

Une carte planaire *enracinée* est une carte planaire dans laquelle un sommet et un arête adjacent sont marqués (Figure 6.4).

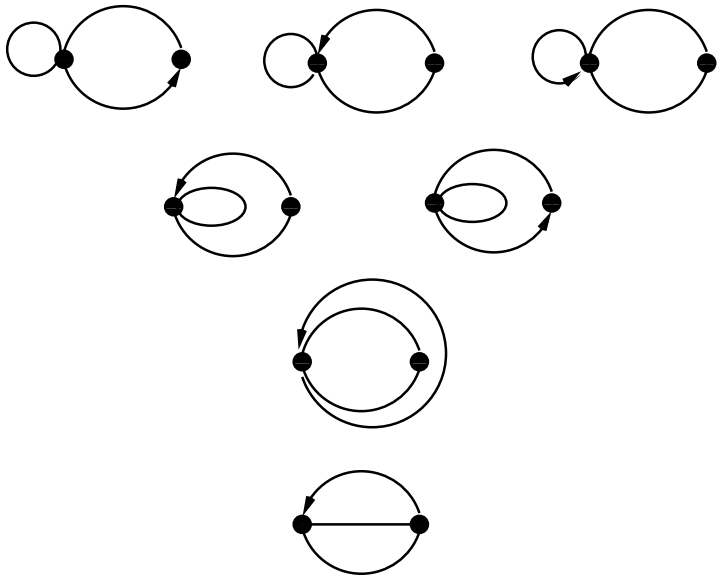
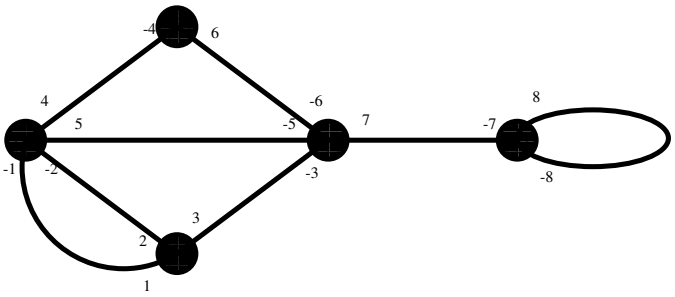


FIGURE 6.4 – ***Les 7 cartes planaires à deux sommets sans isthme enracinées (fléche) à 7 arêtes.***

Toute carte planaire ayant n arêtes peut être représentée par deux permutations σ et τ de l'ensemble $\{+1,-1,+2,-2,\ldots,+(n-1),-(n-1),+n,-n\}$ de la façon

suivante : les arêtes de la carte sont numérotés de 1 à n. Pour chaque arête i, on étiquette une des extrémités par $+i$ et l'autre par $-i$. Par convention, l'arête racine est étiqueté par $+1$ et -1, de telle façon que -1 étiquette l'extrémité adjacente au sommet racine (voir Figure 6.5). Nous avons :

- L'involution sans point fixe σ. Elle représente les arêtes de la carte. Chaque cycle de σ est de taille deux et contient les deux extrémités d'un arête : $\sigma = (+1,-1),(+2,-2),\ldots,(+n,-n)$.
- La permutation τ a autant de cycles qu'il y a de sommets dans la carte. Chaque cycle est donné par la séquence des étiquettes autour du sommet correspondant prises dans le sens des aiguilles d'une montre.



$\sigma = (1,-1)(2,-2)(3,-3)(4,-4)(5,-5)(6,-6)(7,-7)(8,-8)$
$\tau = (1,2,3)(-1,4,5,-2)(-4,6)(-6,7,-3,-5)(-7,8,-8)$

FIGURE 6.5 – ***Exemple de carte planaire et des permutations σ et τ associées.***

Considérons une structure L&P S à n arêtes, et étiquetons l'extrémité gauche de ses arêtes avec $+1,+2,\ldots,+n$ de gauche à droite et donnons à chaque extrémité droite l'étiquette $-i$ si le pied gauche correspondant est étiquetté $+i$. Posons $w = [w_1,w_2,\ldots w_{2n}]$ la séquence des étiquettes de S de gauche à droite.

A partir de tout w, nous pouvons construire deux permutations σ et τ qui représentent une carte planaire enracinée sans isthme à un ou deux sommets. En ce qui concerne σ, nous avons juste l'ensemble $\sigma = (+1,-1)\ldots(+n,-n)$

Considérons d'abord le cas simple où il n'y a pas de pseudonœuds dans la structure. Nous savons depuis longtemps que ces structures empilées sont comptées par les nombres de Catalan. On peut le montrer par exemple par une bijection folklorique avec les cartes planaires à un sommet en posant σ comme sus mentionné et $\tau = (w)$. (voir figure 6.6)

Supposons qu'il y ait un pseudonœud dans la structure. Nous considérons la bijection entre l'ensemble de ces structures et l'ensemble des cartes planaires enracinées sans isthme à deux sommets. Comme τ doit avoir deux cycles, nous prenons le mot w que nous coupons en deux parties. La partie gauche (resp. la partie droite) des arêtes du pseudonœud est associée aux arêtes dont l'extrémité gauche (resp. droite) est dans la partie gauche resp. (droite) du pseudonœud (voir la section 1.1 pour la définition d'un pseudonœud de type H).

On distingue deux cas :

Cas 1. Une seule arête figure dans la partie droite.

Soit ℓ la position de la première extrémité droite dans l'ensemble gauche. La coupure de w a lieu entre les positions $\ell-1$ et ℓ. Chaque partie correspond à un cycle de τ : $\tau = (w_1,\ldots,w_{\ell-1})(w_\ell,\ldots,w_{2n})$. La figure 6.7 illustre cette opération.

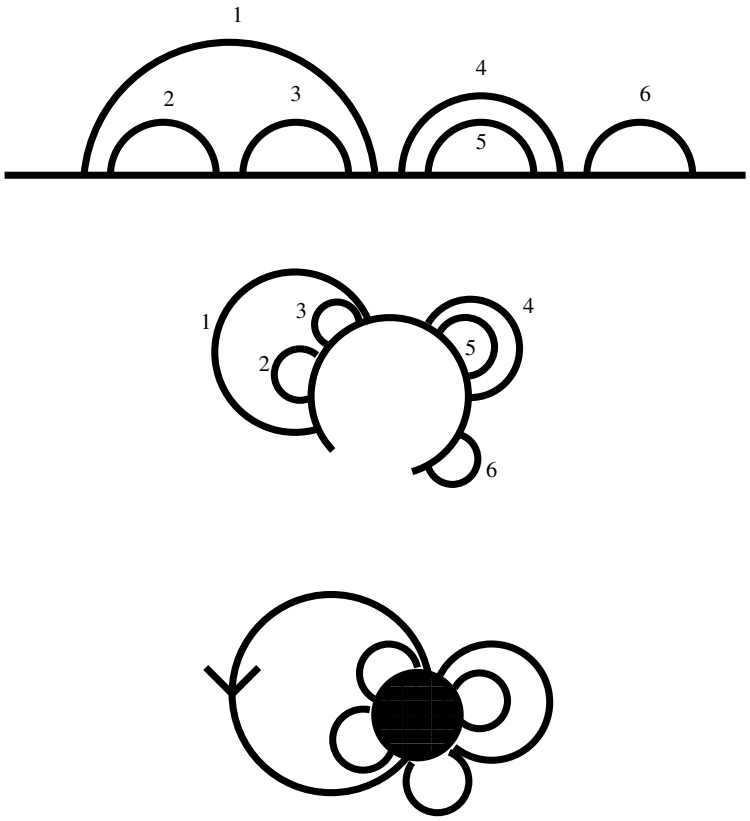


FIGURE 6.6 – ***Bijection folklorique des structures sans pseudonœud vers les cartes planaires à un sommet.***

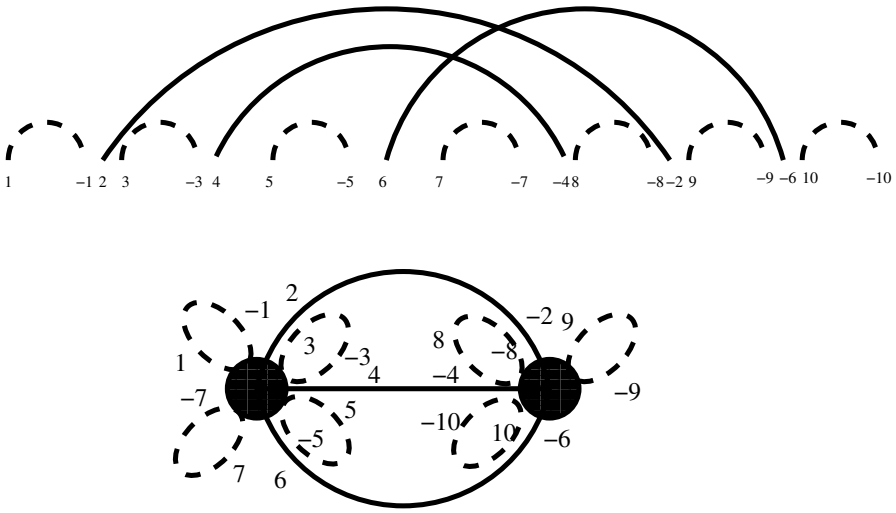


FIGURE 6.7 – ***En haut, une structure L&P correspondant au cas 1. En bas, la carte planaire correspondante. Les arêtes non impliqués dans le pseudonœud sont dessinés en pointillés.***

Cas 2. Au moins deux arêtes figurent dans la partie droite. La coupure de w a lieu juste avant la première extrémité doite d'une arête dans l'ensemble droit. La figure 6.8 illustre ce cas.

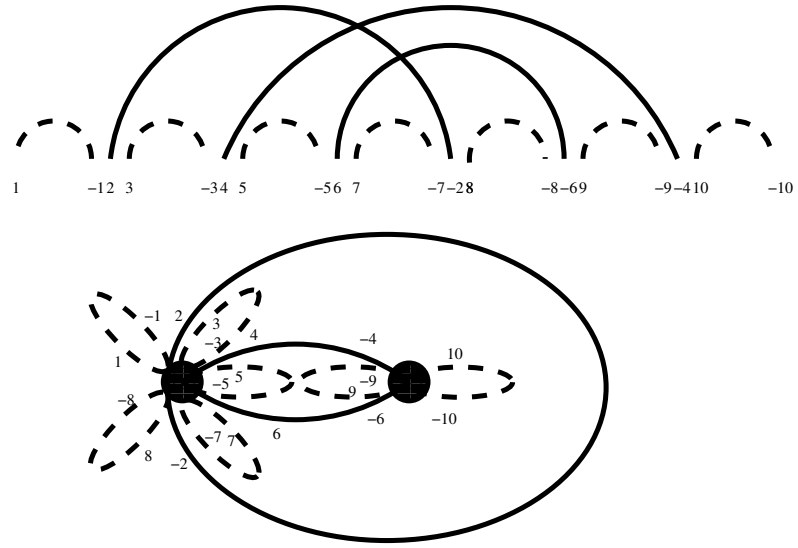


FIGURE 6.8 – ***En haut, une structure L&P correspondant au cas 2. En bas, la carte planaire correspondante. Les arêtes non impliqués dans le pseudonœud sont en pointillés***

Montrons que dans les deux cas, la carte résultante est planaire et sans isthme. On Remarque tout d'abord que si la carte n'est pas planaire ou possède un isthme, cela provient nécesesairement d'une arête impliquée dans le pseudonœud. Il s'en suit par construction que les arêtes non croisantes impliquées dans la structure correspondent à des boucles dans la carte. Nous pouvons donc considérer, sans perte de généralité, que les seules structures où toutes les arêtes sont impliquées dans le pseudonœud. Posons que ces structures aient n arêtes. Dans le cas 1, nous avons :

$$w = [+1, +2, \ldots, +(n-1), +n, -(n-1), -(n-2), \ldots, -1, -n]$$

Il s'en suit

$$\tau = (+1, +2, \ldots, +n)(-(n-1), -(n-2), \ldots, -1, -n).$$

Ce qui nous donne évidemment une carte planaire puisque les deux cycles de τ sont en ordre inverse. Il n'y a pas d'isthme car toutes les arêtes vont d'un sommet à l'autre. Dans le cas 2, nous avons :

$$w = [+1, +2, \ldots, +(\ell-1), +\ell, +(\ell+1), \ldots, +n, -(\ell-1), \ldots, -2, -1, -n, -(n-1), \ldots, -\ell]$$

dès lors

$$\tau = (+1, +2, \ldots, +(\ell-1), +\ell, +(\ell+1), \ldots, +n, -(\ell-1), \ldots, -2, -1)(-n, -(n-1), \ldots, -\ell).$$

Nous obtenons à nouveau une carte planaire : $1, 2, \ldots, \ell-1$ sont des boucles imbriquées, et les arêtes $\ell, \ldots, n$ vont d'un sommet à l'autre sans se croiser. Il n'y a pas d'isthme car le nombre de sommets allant d'un sommet vers l'autre $n-\ell+1$, est supérieur ou égal à 2.

La transformation réciproque se présente de la façon suivante. On considère la carte planaire enracinée sans isthme à deux sommets définie par :$\sigma = (+1,-1),(+2,-2),\ldots,(+n,-n)$ et τ ayant deux cycles. Nous construisons la séquence w qui représente la structure pseudonouée correspondante. Conséidérons le cycle τ qui contient 1, et écrivons le de telle façon qu'il commence par 1. On appelle u cette séquence d'étiquettes. On obtient ainsi la première partie de la séquence w. Recherchons maintenant la seconde partie de w, c'est à dire la séquence v telle que $w = uv$. Dans ce but, On considére l'ensemble des *étiquettes isolées*, telles que les étiquettes de u n'aient pas leur étiquette opposée dans u. Il s'en suit :

1. Il n'y a pas de paire $(+i,-i)$ dans u tel que les sommets isolés se trouvent entre $+i$ et $-i$. Soit $+j$ la pénultième étiquette isolée de u. Ecrivons le second cycle τ de telle manière qu'il commence par $-j$. Cela nous donne v, il y a alors exactement un arête dans la partie gauche du pseudonœud.
2. Il y a une paire d'étiquettes $+i,-i$ dans u telles que toutes les étiquettes isolées se trouvent entre $+i$ et $-i$. Posons $+j$ la dernière étiquette isolée de u. On écrit le second cycle de τ de telle façon qu'il commence par $-j$. Cela donne v. Dans ce cas, il y a au moins deux sommets dans la seconde partie du pseudonœud.

□

2.2 La classe des UPK et arbres ternaires

Nous avons précédemment montré une méthode pour dénombrer les structures de la classe *L&P*. Nous avons également vu dans la section 1.1 que nous pouvions étudier les croisements des pseudonœuds indépendamment de leur position dans la structure. Dans cette section, nous allons étudier le problème de la position en définissant une classe d'équivalence, les pseudonœuds indifférenciés.

Définitions préliminaires

Définition 43 (Pseudonœuds équivalents)**.** *Deux pseudonœuds P et P' sont équivalents si et seulement si leurs graphes de cohérence (voir section 1.1) ont la même taille. Nous notons alors $P \equiv P'$.*

Définition 44 (Structures pseudonouées équivalentes)**.** *Soit S (resp. S') une structuture secondaire avec $P = \{P_1,\ldots,P_n\}$ l'ensemble des pseudonœuds de S (resp. $P' = \{P'_1,\ldots,P'_n\}$ l'ensemble des pseudonœuds de S')*

S et S' sont equivalentes si :

- *Les structures ont la même taille.*
- *Les extrémités des structures non croisées de S et de S' ont les mêmes coordonnées.*
- *$\forall 1 \leq k \leq n, P_k\ P'_k$ et leurs extrémités ont les mêmes coordonnées.*

Nous notons cette classe d'équivalence UPK pour Undifferentiated PseudoKnoted structures.

Décrivons plus précisément ces structures dans le but de les énumérer. Cette description devient naturelle si nous montrons comment transformer une structure secondaire en UPK.

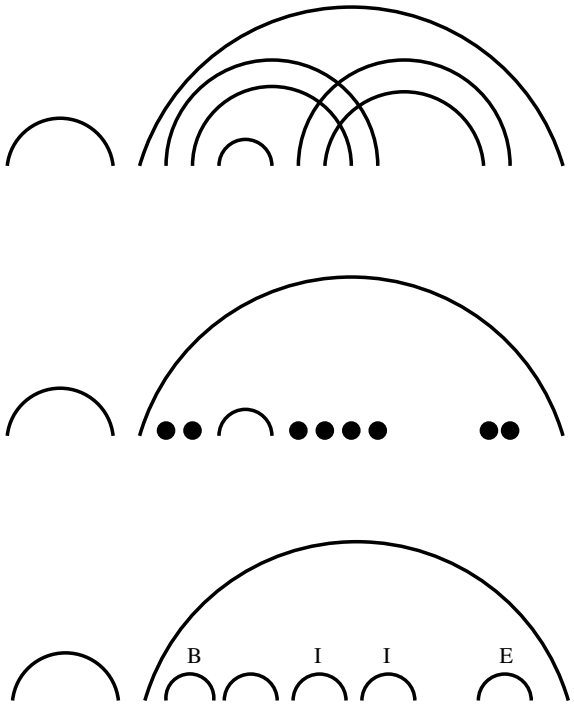


FIGURE 6.9 – ***En haut : Structure secondaire. Milieu :*** UPK ***correspondant. Bas : Codage des UPK par des arcs étiquetés.***

La fonction f suivante décrit cette tranformation. Si un arc n'est pas impliqué dans un pseudonœud, il demeure inchangé. On a alors :

$$f: \quad (i_k, j_k) \notin P \quad \rightarrow \quad (i_k, j_k)$$

Les arcs impliqués dans un même pseudonoœud, sont remplacés par leur position seule. On a alors

$$((i_1, j_1), \ldots (i_k, j_k)) \in P, k \geq 2 \qquad \{i_1, j_1, \ldots i_k, j_k\}$$

Dénombrement de la classe d'équivalence UPK

Théorème 18. *Le nombre de structures UPK ayant n arcs est* $f(n) = \binom{3n}{n} \frac{1}{2n+1}$.

Nous donnons deux façons différentes de prouver ce théorème. La première méthode consiste à montrer que les UPK sont en bijection avec les arbres ternaires.

Dans la seconde, nous codons les UPK par des mots décrits par une grammaire non contextuelle. Nous énumérons alors les mots produits par cette grammaire.

Bijection entre UPK et arbres ternaires. Montrons d'abord la correspondance entre les structures UPK et les arbres ternaires

On code les UPK de la façon suivante :

- Pour chaque pseudonœud $P_k, 1 \leq k \leq n$.
  - Trions par ordre croissant les extrémités impliquées dans P_k. On obtient la séquence triée $i_1 \ldots i_{|P_k|}$
  - Pour chaque couple $i_l, i_{l'}, l$ est impair on associe un arc. (Figure 6.9)

  - On étiquette le premier arc par B. On dit que cet arc débute le pseudonœud indifférencié.
  - On étiquette le dernier arc par E. On dit que cet arc termine le pseudonœud indifférencié.
  - On étiquette les autres arcs par I. On dit que ces arcs continuent le pseudonœud indifférencié.
- On garde les arcs non impliqués dans un pseudonœuds. Ils ne reçoivent pas d'étiquette.

On appelle S' la structure résultante.

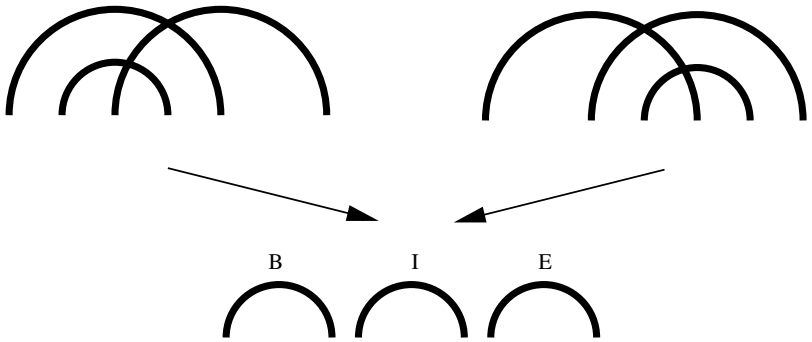


FIGURE 6.10 – ***Deux structures avec pseudonoeud (en haut) avec le même pseudonœud indifférencié (en bas).***

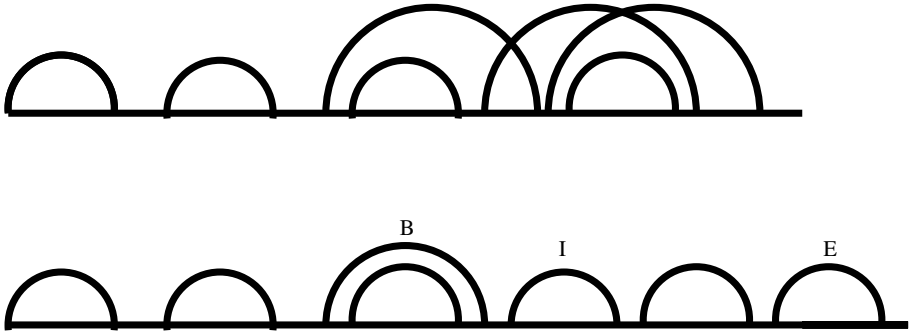


FIGURE 6.11 – ***En haut, structure secondaire avec pseudonœud, en bas une structure secondaire avec pseudonœud indifférencié et son codage.***

Il n'y a pas d'arc croisant dans S' (Figures 6.11 et 6.10)

Nous construisons l'arbre ternaire à partir de la structure UPK de la façon suivante :

La structure secondaire avec pseudonœud indifférencié peut être décomposée en arcs voisins ou empilés (Figure 6.12).

Soit un arc U.

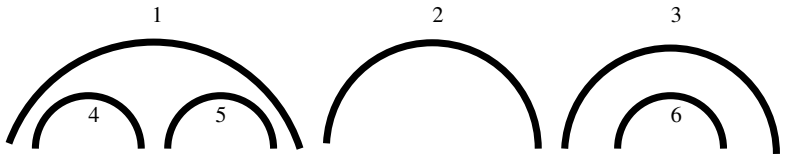


FIGURE 6.12 – ***4 et 6 sont les premiers arcs respectivement empilés sous 1 et 3. 2, 5 et 3 sont les premiers arcs respectivements voisins de 1, 4 et 2.***

Si U est le premier arc de la structure, on dessine la racine de l'arbre.

Si V est le premier arc empilé sous U après l'extrémité gauche de U (Figure 6.13 haut) alors on dessine un fils gauche à u, le noeud associé à U dans l'arbre ternaire.

Si V est le premier voisin de U et que V est non étiqueté ou étiqueté B, alors on dessine un fils centre à u, le noeud correspondant à U dans l'arbre ternaire (Figure 6.13 centre)

Sinon U est étiqueté B ou I et V est l'arc qui continue ou termine le pseudonœud indifférencié. (Figure 6.13 bas)

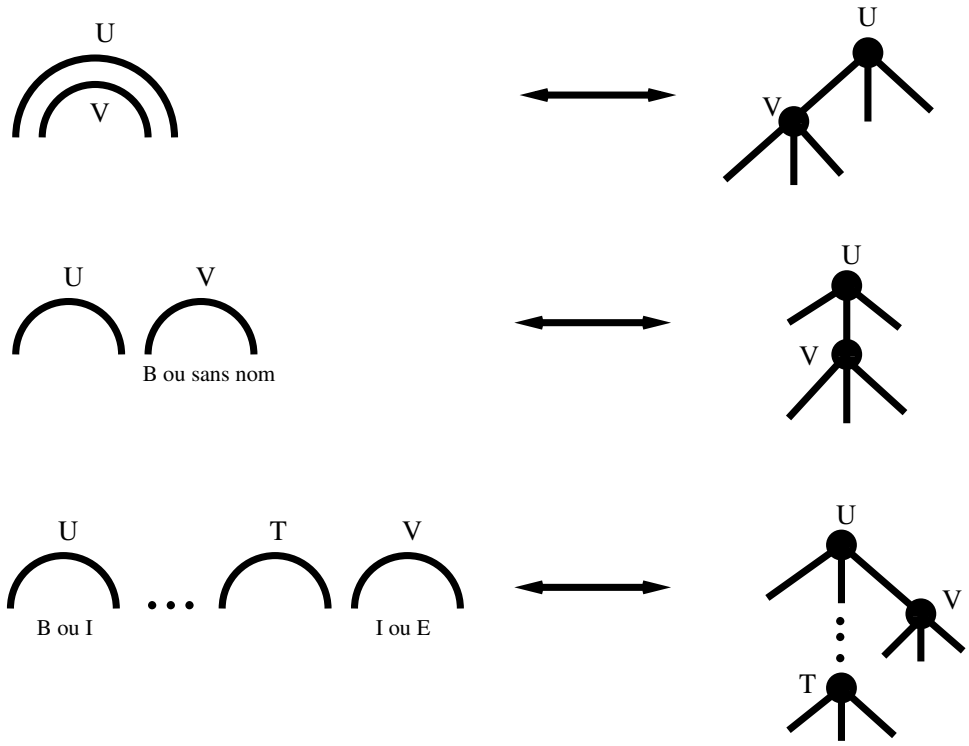


FIGURE 6.13 – ***En haut, structures empilées. Milieu, structures voisines n'étendant pas un pseudonœud. En bas, structures voisines étendant un pseudonœud.***

Nous avons montré que pour chaque structure, nous pouvons construire l'arbre ternaire correspondant. Nous devons montrer maintenant comment construire une structure avec pseudonœud indifférencié en partant d'un arbre ternaire.

L'algorithme suivant réalise la bijection inverse :
- Racine : Dessiner un arc.
  - S'il a un fils droit, étiquetons l'arc correspondant avec B.
  - Sinon on ne met pas d'étiquette.
- Pour chaque arc u dans l'ordre du parcour en profondeur main gauche.
  - Si u a un fils gauche, dessine un arc juste après l'extrémité gauche de l'arc correspondant au père.
    - Si u a un fils droite, on étiquette l'arc correspondant avec B.
    - Sinon, on n'étiquette pas l'arc.
  - Si u est un fils centre, on dessine un arc juste après l'extrémité droite de son père.
    - Si u a un fils droite, on étiquette l'arc correspondant avec B.
    - Sinon, on n'étiquette pas l'arc.
  - Si u est un fils droite on dessine un arc à l'extrémité droite.
    - Si u a un fils droit, étiquetons l'arc correspondant I.
    - Sinon, étiquettons l'arc correspondant E.

□

Nous avons démontré le théorème par une preuve bijective. Donnons une autre façon de prouver le théorème en utilisant une gramaire non contextuelle.

Démonstration. Nous codons la structure secondaire par un mot. Construisons-le de la manière suivante :
- L'extrémité gauche des arcs non étiquetés est nommée d. La droite $\bar{d}$.
- L'extrémité gauche des arcs étiquetés B est nommée x. La droite a
- L'extrémité gauche des arcs étiquetés I est nommée a. La droite a
- L'extrémité gauche des arcs étiquetés E est nommée a. La droite $\bar{x}$

On lit la séquence de lettres de gauche à droite, ce qui forme un mot décrit par la grammaire non contextuelle suivante :

$$S \to dS\bar{d}S|P$$

$$P \to xSaSC|\varepsilon$$

$$C \to aSaSC|aS\bar{x}S$$

On en déduit le système d'équation suivant selon la méthode DSV décrite dans la section 1 :

$$S(z) = z^2S^2(z) + P(z)$$

$$P(z) = z^2S^2(z)C(z) + 1$$

$$C(z) = z^2S^2(z)C(z) + z^2S^2(z)$$

On réduit aisément le système et on obtient :

$S(z) = z^2 S^3(z) + 1$

Si nous considérons, $Z = z^2$ le nombre d'arcs, on obtient :

$S(Z) = ZS^3(Z) + 1$ est l'équation pour la série génératrice des arbres ternaires. □

La figure 6.14 résume les transformations précedentes.

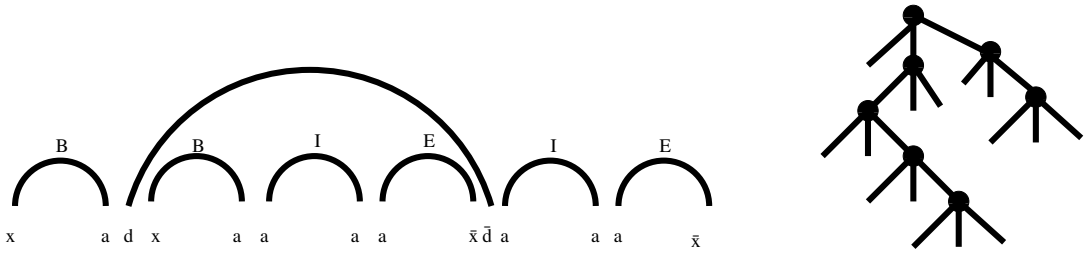


FIGURE 6.14 – ***Arbre ternaire (droite) et la structure secondaire correspondante avec pseudonœuds indifférenciés (gauche).***

3 Génération aléatoire de séquences avec pseudonœuds

3.1 Algorithmes en temps linéaire pour reconstruire les pseudonœuds

Nous présentons tout d'abord un algorithme qui prend en entrée un mot u qui représente un pseudonœud simple unique, et qui construit le pseudonœud. En d'autres termes, l'algorithme construit $f^{-1}(u)$, où f est la fonction réalisant le codage décrit dans la proposition 2 dans la section 1.1. Les mots sont considérés comme des tableaux de caractères. L'algorithme se déroule en deux étapes :

1. Lecture du mot de gauche à droite et écriture des lettres dans le même ordre, sauf les $\bar{y}$. On remplace respectivement les p et $\bar{p}$ par x et $\bar{x}$.
2. Les $\bar{y}$ sont écrits en fin de mot.

un mot u de taille n qui code pour un pseudonœud $u \leftarrow f^{-1}(u)$ $j \leftarrow 1$ $i = 1$ à n $u_i \in \{x, \bar{x}, y\}$ $u_j \leftarrow u_i$ $j \leftarrow j+1$ $u_i = p$ $u_j \leftarrow x$ $j \leftarrow j+1$ $u_i = \bar{p}$ $u_j \leftarrow \bar{x}$ $j \leftarrow j+1$ $k = j$ à n $u_k \leftarrow \bar{y}$

Algorithm 1: Construction d'un pseudonœud simple à partir de son codage non croisé

Maintenant nous pouvons écrire l'algorithme 2 qui prend en entrée le codage de u d'une structure récursivement pseudonouée, et qui donne le pseudonœud $f^{-1}(u)$. Comme tout pseudonœud peut contenir une autre structure pseudonouée, une pile est utilisée. Chaque élément de la pile contiendra la liste des positions des sommets d'un pseudonœud donné dans la structure. Quand la fin du codage d'un pseudonœud est atteinte, il est dépilé et la procédure *CroiseSousMot* est appellée (Voir algorithme 3). Cette procédure est assez similaire à celle de l'algorithme 1, la seule différence est que la position des sommets du pseudonœud dans le mot u qui contient le pseudonœud est prise en compte.

un mot u de taille n $u \leftarrow f^{-1}(u)$ $i = 1$ à n [Début d'un pseudonœud]$u_i = p$ Créer une liste vide L Empiler i sur L [A l'intérieur d'un pseudonœud]$u_i \in \{x, \bar{x}, y, \bar{y}\}$ Empiler i sur L [Fin d'un pseudonœud]$u_i = \bar{p}$ CroiseSousMot(u,L)

Algorithm 2: Construction d'une structure avec pseudonœuds simples récursifs à partir de son codage non croisé.

un mot u de taille n, une liste $L = (q_1, q_2, \ldots, q_m)$ de positions dans u $u_{q_1}u_{q_2}\ldots u_{q_m} \leftarrow f^{-1}(u_{q_1}u_{q_2}\ldots u_{q_m})$ $j \leftarrow 1$ $i = 1$ to m $u_{q_i} \in \{x, \bar{x}, y\}$ $u_{q_j} \leftarrow u_{q_i}$ $j \leftarrow j+1$ $u_{q_i} = p$ $u_{q_j} \leftarrow x$ $j \leftarrow j+1$ $u_{q_i} = \bar{p}$ $u_{q_j} \leftarrow \bar{x}$ $j \leftarrow j+1$ $k = j$ à n $u_{q_k} \leftarrow \bar{y}$

Algorithm 3: Procédure CroiseSousMot pour les pseudonœuds simples

Pour les deux autres types de pseudonoeuds que nous allons traiter dans la section 3.2 nous adopterons la même stratégie, seule la procédure *CroiseSousMot* change. Nous savons donc engendrer aléatoirement de façon uniforme ou non uniforme controlée en $O(nlog(n))$ où n est la taille de la séquence engendrée *e.g.* [43, 35, 84] les structures prédictibles par les algorithmes *ab initio* exacts.

3.2 Exemples de structures aléatoires avec pseudonœuds

A partir des grammaires non contextuelles non ambiguës décrites précédemment nous savons engendrer de façon uniforme ou pondérée [30] des structures aléatoires d'ARN dont le codage correspond à des pseudonœuds non croisés à l'aide de GENRGENS [84]. Grâce aux algorithmes décrits en 3.1, nous pouvons reconstituer les pseudonœuds sans surcoût de complexité.

Nous avons engendré des structures aléatoires correspondant à des structures simplement pseudonouées que nous présentons dans la figure 6.15.

On peut également engendrer des structures plus réalistes à l'aide de grammaires plus complexes en favorisant par exemple des hélices de plus longue taille :

$S \rightarrow T|cS|P$

$T \rightarrow N|Q$

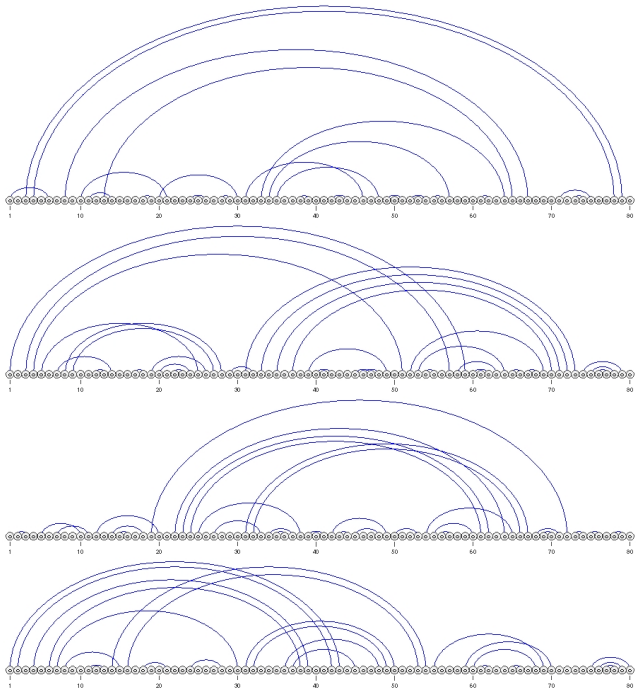


FIGURE 6.15 – ***Quatre structures aléatoires engendrées avec la grammaire correspondant aux structures de la classe*** *D&P*.

$N \to dN\bar{d}\bar{S}|QT$

$Q \to dQ\bar{d}|ddd\,ccc\bar{d}\bar{d}\bar{d}$

$P \to pxxXS\bar{x}\bar{x}\bar{p}S|\varepsilon$

$X \to xSX\bar{x}S|yyySY\bar{y}\bar{y}\bar{y}S$

$Y \to ySY\bar{y}S|\varepsilon$

On peut également modifier les pondération afin d'engendrer plus de pseudonœuds que dans le modèle uniforme (Figure 6.16).

4 Prolongements

Dans cette partie, nous avons étudié les différents algorithmes de prédiction de structure d'ARN *ab initio* exacts afin de caractériser précisément les pseudonœuds qu'ils prédisent (section 2). Puis nous avons étudié le compromis entre l'espace de prédiction et la complexité en temps des algorithmes (section 1) grâce à une transformation bijective qui nous permet de décrire les structures simplement pseudonouées par des mots appartenant à des langages non contextuels. Nous avons donné dans la section 3.1 un algorithme en temps linéaire qui réalise cette transformation, ce qui nous permet d'engendrer aléatoirement des structures pseudonouées correspondant aux différentes classes sans surcoût de complexité par rapport aux struc-

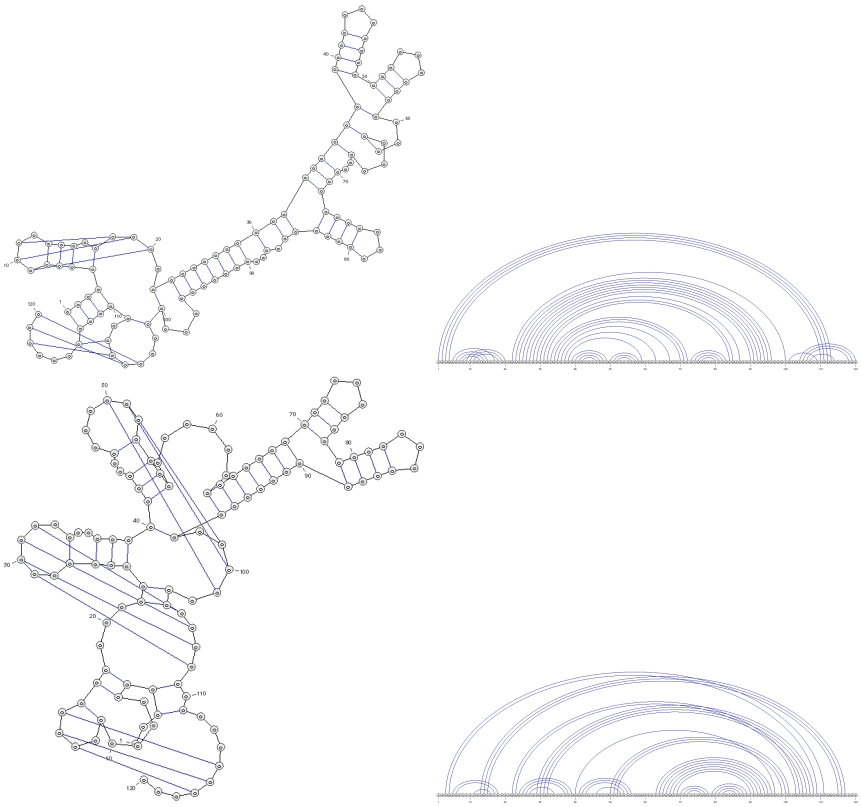


FIGURE 6.16 – ***Deux structures aléatoires engendrées à l'aide d'une grammaire pondérée. A gauche : représentation en deux dimensions. A droite : la représentation arc-annotée correspondante.***

tures sans pseudonœud. Nous pouvons désormais développer des modèles aléatoires qui prennent en compte les pseudonœuds. Ces modèles permettraient de calculer la Z-valeur pour des algorithmes de comparaison de structures avec pseudonœuds [52, 76]. Bien que nous n'ayons pas de codage pour les structures prédictibles par l'algorithme de *R&E*, ni pour les structures *G&D*, nous pouvons décrire certaines structures spécifiques de ces algorithmes comme les épingles à cheveux embrassées et certaines structures non planaires (voir section 1.1).

Codage des épingles à cheveux embrassées

Lemme 10. *Toute classe de structures pseudonouées où tous les pseudonœuds sont des épingles à cheveux embrassées peut être codées par les mots du langage L sur l'alphabet* $\{d, \bar{d}, x, \bar{x}, y, \bar{y}, \}$ *où :*

- *d et* $\bar{d}$ *codent, respectivement, les extrémités gauches et droites des arcs qui ne sont pas impliquées dans un pseudonœud ;*
- *x et* $\bar{x}$ *codent, respectivement, les extrémités gauches et droites des arcs impliquées dans un pseudonœud. Ces arcs ne se coupent pas entre eux. Ils coupent les arcs de type* $\{y, \bar{y}\}$ *;*
- *y et* $\bar{y}$ *codent, respectivement, les extrémités gauches et droites des arcs impliquées dans un pseudonœud. Ces arcs ne se coupent pas entre eux. Ils coupent les arcs de type* $\{x, \bar{x}\}$ *;*

– *c code les bases non appariées ;*

De plus, la projection du langage sur l'alphabet $\{d,\bar{d}\}$ *(resp.* $\{x,\bar{x}\}$, $\{y,\bar{y}\}$*) est un sous langage de Dyck sur le même alphabet.*

Lemme 11. *Soit S une structure pseudonouée, et w le mot sur* $\{d,\bar{d},x,\bar{x},y,\bar{y},c\}$ *qui code S. Alors, chaque structure en forme d'épingle à cheveux embrassées de S est codée par un sous-mot v de w, tel que*

$$v = x^{n_1} y^m \bar{x}^{n_1} x^{n_2} \bar{y}^m \bar{x}^{n_2},$$

On peut généraliser ce codage à un nombre k d'épingles à cheveux embrassées.

Lemme 12. *Posons S une structure pseudonouée, et w le mot sur* $\{d,\bar{d},x,\bar{x},y,\bar{y},c\}$ *qui code S. Alors, chaque structure en forme de k épingles à cheveux embrassées de S est codée par un sous-mot v de w, tel que*

$$v = x^{n_1} y^{m_1} \bar{x}^{n_1} x^{n_2} \bar{y}^{m_1} y^{m_2} \bar{x}^{n_2} \ldots x^{n_k} \bar{y}^{m_k-1} \bar{x}^{n_k}$$

Proposition 3. *Posons S comme une structure pseudonouée, et w comme un mot sur* $\{d,\bar{d},x,\bar{x},y,\bar{y},c\}$ *qui code S. Alors w peut être codé par un mot où chaque sous-mot :*

$$v = x^{n_1} y^{m_1} \bar{x}^{n_1} x^{n_2} \bar{y}^{m_1} y^{m_2} \bar{x}^{n_2} \ldots x^{n_k} \bar{y}^{m_k-1} \bar{x}^{n_k}$$

correspondant à un pseudonœud composé de k épingles à cheveux embrassées est remplacé par :

$$v' = p\, x^{n_1-1} y^{m_1} x^{n_2} y^{m_2} \ldots y^{m_k-1} x^{n_k} \bar{x}^{n_k} \bar{y}^{m_k-1} \ldots \bar{y}^{m_2} \bar{x}^{n_2} \bar{y}^{m_1} \bar{x}^{n_1-1} \bar{p}$$

Démonstration. Similaire à la proposition 2. La transformation est illustrée dans la figure 4.

□

L'algorithme 4 décrit la tranformation décrite dans la proposition 3. Le principe de l'algorithme repose sur l'utilisation de piles, la première pile compte le nombre de lettres $\bar{x}$ nécessaires et la seconde le nombre de $\bar{y}$. Le codage donné dans la proposition 3 fait que nous écrivons autant de $\bar{x}$ qu'il y en a dans la pile lorsque l'on rencontre le premier x ou p d'une suite de pieds gauche d'arcs. On fait de même pour les y.

un mot u de taille n, une liste $L = (q_1, q_2, \ldots, q_m)$ de positions dans u
$u_{q_1} u_{q_2} \ldots u_{q_m} \leftarrow f^{-1}(u_{q_1} u_{q_2} \ldots u_{q_m})$ Créer liste vide L_1 Créer liste vide L_2
$j \leftarrow 1$ $i \leftarrow 1$ $i < m$ $u_{q_i} \in \{x,p\}$ L_1 est_non_vide() $u'_j \leftarrow \bar{x}$ Dépiler L_1 $j \leftarrow j+1$
$u_{q_i} = x$ $u'_j \leftarrow x$ Empiler $\bar{x}$ sur L_1 $i \leftarrow i+1$ $j \leftarrow j+1$ $u_{q_i} = y$ L_2 est_non_vide()
$u'_j \leftarrow \bar{y}$ Dépiler L_2 $j \leftarrow j+1$ $u_{q_i} = y$ $u'_j \leftarrow y$ Empiler $\bar{y}$ sur L_1 $i \leftarrow i+1$
$j \leftarrow j+1$ $i = 1$ à m $u_{q_i} \leftarrow u'_i$

Algorithm 4: Procédure CroiseSousMot pour les épingles à cheveux embrassées

Cet algorithme permet donc d'obtenir des structures pseudonouées dans lesquelles les pseudonœuds sont des épingles à cheveux embrassées à partir de leur codage non croisé en temps linéaire.

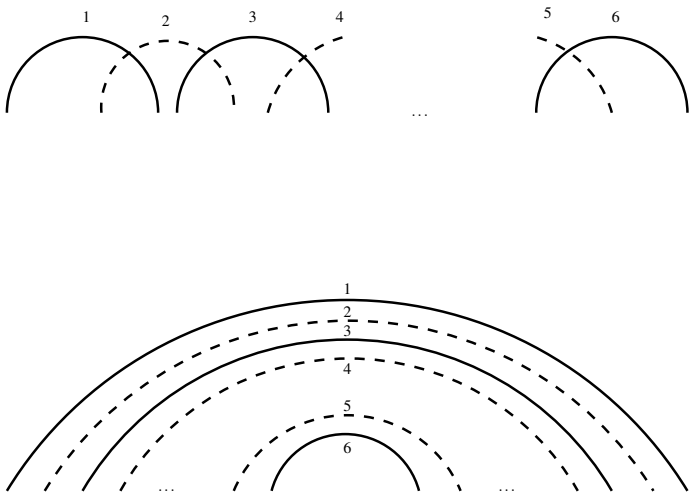


FIGURE 6.17 – ***Pseudonœud non planaire. En bas la structure codée par v' selon la proposition 3. Les lignes pleines représentent*** x ***et*** $\bar{x}$ ***et*** p ***et*** $\bar{p}$***, les lignes pointillées représentent*** y ***et*** $\bar{y}$***.***

Codage des pseudonœuds non planaires dont le graphe de cohérence est K_n
Nous allons procéder de la même manière pour les pseudonœuds non planaires. Ces structures non planaires sont celles dont le graphe de cohérence est une clique. Elles sont spécifique à la classe *R&E*. Pour ces structures, la taille de l'alphabet est égale à la taille de la plus grande clique du graphe de cohérence +2 (pour gérer les bases non impliquées dans un pseudonœud et les bases non appariées).

Lemme 13. *Toute classe de structures pseudonouées où tous les pseudonœuds sont non planaires peuvt être codées par les mots du langage L sur l'alphabet* $\{z_1, \bar{z}_1, \ldots, z_k, \bar{z}_k, c\}$ *où :*

- *d et* $\bar{d}$ *codent, respectivement, les extrémités gauches et droites des arcs qui ne sont pas impliquées dans un pseudonœud ;*
- z_1 *et* $\bar{z}_1$ *codent, respectivement, les extrémités gauches et droites des arcs impliquées dans un pseudonœud. Ces arcs ne se coupent pas entre eux. Ils coupent tous les autres arcs sauf* $\{d, \bar{d}\}$ *;*
- ...
- z_k *et* $\bar{z}_k$ *codent, respectivement, les extrémités gauches et droites des arcs impliquées dans un pseudonœud. Ces arcs ne se coupent pas entre eux. Ils coupent tous les autres arcs sauf* $\{d, \bar{d}\}$ *;*
- *c code les bases non appariées ;*

De plus, la projection du langage sur l'alphabet $\{d, \bar{d}\}$ *(respectivement* $\{z_1, \bar{z}_1\}$*,* ..., $\{z_k, \bar{z}_k\}$*) est un sous-langage de Dyck sur le même alphabet.*

Lemme 14. *Soit S une structure pseudonouée, et w le mot sur* $\{d, z_1, \bar{z}_1, \ldots, z_k, \bar{z}_k, c\}$ *qui code S. Alors, chaque structure non planaire de S est codée par un sous-mot v de w, tel que*

$$v = {z_1}^{n_1} {z_2}^{n_2} {z_3}^{n_3} {z_4}^{n_4} {z_5}^{n_5} \ldots {z_{k-1}}^{n_{k-1}} {z_k}^{n_k} \ldots \bar{z}_1^{n_1} \bar{z}_2^{n_2} \bar{z}_3^{n_3} \bar{z}_4^{n_4} \bar{z}_5^{n_5} \ldots \bar{z}_{k-1}^{n_{k-1}} \bar{z}_k^{n_k}$$

Proposition 4. *Posons S comme une structure pseudonouée, et w comme un mot sur* $\{d, \bar{d}, z_1, \bar{z}_1, \ldots, z_k, \bar{z}_k, c\}$ *qui code S. Alors w peut être codé par un mot où chaque sous-mot :*

$$v = {z_1}^{n_1} {z_2}^{n_2} {z_3}^{n_3} {z_4}^{n_4} {z_5}^{n_5} \ldots {z_{k-1}}^{n_{k-1}} {z_k}^{n_k} \bar{z}_1^{n_1} \bar{z}_2^{n_2} \bar{z}_3^{n_3} \bar{z}_4^{n_4} \bar{z}_5^{n_5} \ldots \bar{z}_{k-1}^{n_{k-1}} \bar{z}_k^{n_k}$$

est transformé en :

$$v' = p\, {z_1}^{n_1-1} {z_2}^{n_2} {z_3}^{n_3} {z_4}^{n_4} {z_5}^{n_5} \ldots {z_{k-1}}^{n_{k-1}} {z_k}^{n_k} \bar{z}_k^{n_k} \bar{z}_{k-1}^{n_{k-1}} \ldots \bar{z}_5^{n_5} \bar{z}_4^{n_4} \bar{z}_3^{n_3} \bar{z}_2^{n_2} \bar{z}_1^{n_1-1}\, \bar{p}$$

Démonstration. Similaire à la proposition 2. La transformation est illustrée dans la figure 4.

□

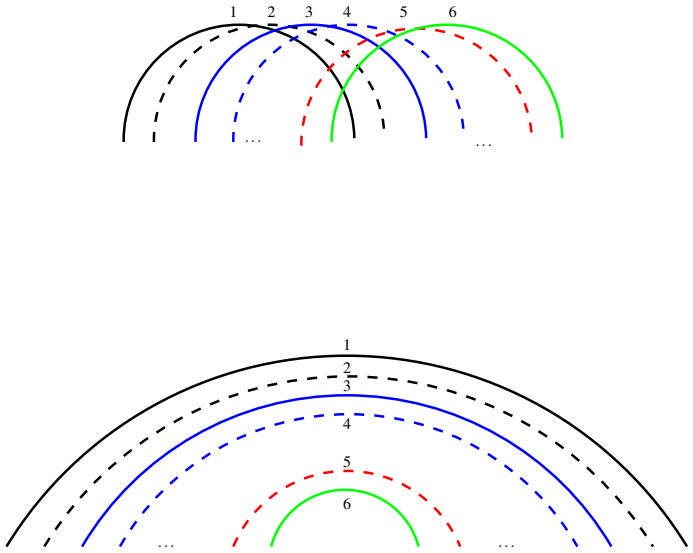


FIGURE 6.18 – ***En haut : pseudonœud non planaire. En bas, la structure codée par v' selon la proposition 4.***

Théorème 19. *L'ensemble des épingles à cheveux contenant n arcs et des structures non planaires dont le graphe de cohérence est* K_n *ont la même taille.*

Démonstration. Considérons le codage des épingles à cheveux embrassées lemme 10 et augmentons la taille de l'alphabet de la façon suivante :

$$v = x^{n_1} y^{m_1} \bar{x}^{n_1} x^{n_2} \bar{y}^{m_1} y^{m_2} \bar{x}^{n_2} \ldots x^{n_k} \bar{y}^{m_k-1} \bar{x}^{n_k}$$

devient :

$$v' = {z_1}^{n_1} {z_2}^{m_1} \bar{z}_1^{n_1} {z_3}^{n_2} \bar{z}_2^{m_1} {z_4}^{m_2} \bar{z}_3^{n_2} \ldots {z_k}^{n_k} \bar{z}_{k-1}^{m_{k-1}} \bar{z}_k^{n_k}$$

Si nous considérons la proposition 3 et que l'on applique l'augmentation de taille de l'alphabet précédemment décrite sur la structure non croisée, nous obtenons le codage suivant :

$$v' = p\, z_1{}^{n_1-1}\, z_2{}^{m_1}\, z_3{}^{n_2}\, z_4{}^{m_2} \ldots z_k{}^{n_k}\, \bar{z}_k^{n_k} \ldots \bar{z}_4^{m_2}\, \bar{z}_3^{n_2}\, \bar{z}_2^{m_1}\, \bar{z}_1^{n_1-1}\, \bar{p}$$

On reconnaît là le codage non croisé des structures non planaires décrit dans la proposition 4. Cette transformation est illustrée en figure 4.

□

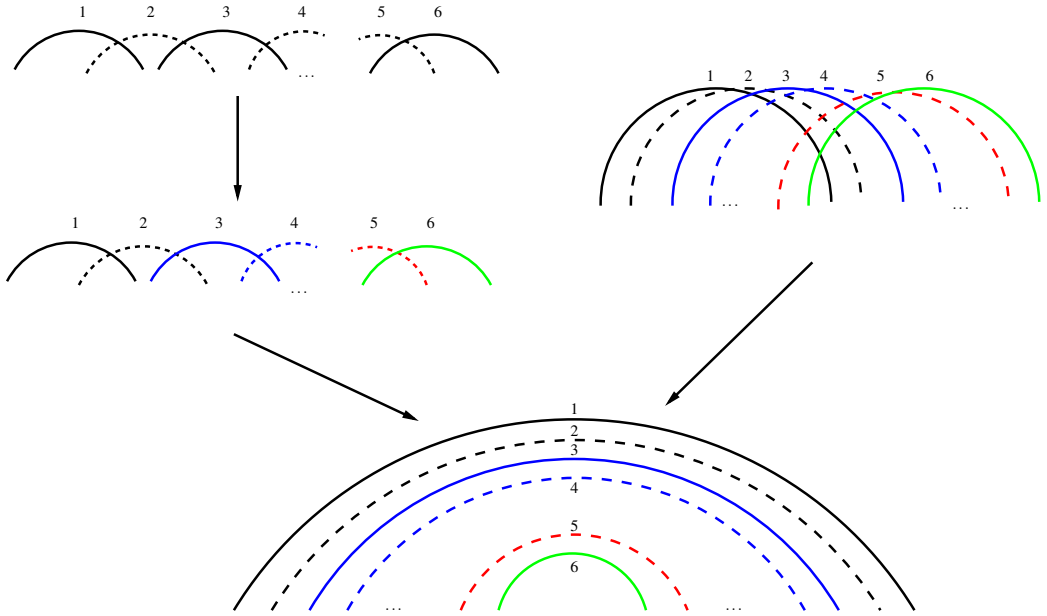


FIGURE 6.19 – ***Illustration de la bijection de l'ensemble des épingles à cheveux embrassées (à gauche) avec les pseudonœuds non planaires (à droite). A gauche on peut voir l'illustration de l'augmentation de taille de l'alphabet pour les épingles à cheveux embrassées qui est à la base de la mise en correspondance. En bas, on voit que la structure non croisée est la même pour les deux ensembles de structures.***

Dans cette procédure, nous devons tenir compte du nombre de lettres qui décrivent les structures. L'algorithme 5 décrit la transformation décrite dans la proposition 4 consiste en deux étapes :

1. Ecrire les lettres correspondant aux pieds gauches $(z_1, \ldots, z_k)$ dans l'ordre de leur apparition.
2. Ecrire les lettres correspondant aux pieds droits $(\bar{z}_1, \ldots, \bar{z}_k)$ dans l'ordre inverse de leur apparition.

un mot u de taille n, une liste $L = (q_1, q_2, \ldots, q_m)$ de positions dans u
$u_{q_1} u_{q_2} \ldots u_{q_m} \leftarrow f^{-1}(u_{q_1} u_{q_2} \ldots u_{q_m})$ $j \leftarrow 1$ $i = 1$ to m $u_{q_i} \in \{z_1, \ldots, z_k\}$
$u_{q_j} \leftarrow u_{q_i}$ $j \leftarrow j+1$ $u_{q_i} = p$ $u_{q_j} \leftarrow x$ $j \leftarrow j+1$ $l \leftarrow j$ $k = m$ à j
$u_{q_k} \in \{\bar{z}_1, \ldots, \bar{z}_k\}$ $u_{q_l} \leftarrow u_{q_k}$ $l \leftarrow l-1$ $u_{q_k} = \bar{p}$ $u_{q_l} \leftarrow \bar{x}$

Algorithm 5: Procédure CroiseSousMot pour les pseudonœuds non planaires

Les algorithmes précédents ont tous une complexité en temps linéaire pour coder les bijections. Nous savons donc engendrer aléatoirement de façon uniforme ou non uniforme controlée en $O(nlog(n))$ où n est la taille de la séquence engendrée *e.g.* [43, 35, 84] la plupart structures secondaires d'ARN qui ont biologiquement du sens [39].

Conclusions générales

Au cours de ce travail, après avoir introduit les notions afférentes à cette recherche (sur les structures d'ARN, la théorie des langages et la génération aléatoire), nous nous sommes intéressé, dans un premier temps, au problème de la significativité des scores de comparaison de structures secondaires d'ARN sans pseudonœud et, dans un deuxième temps, nous avons étudié les algorithmes exacts de prédiction de structure secondaire d'ARN avec pseudonœuds.

Nous avons développé dans la première partie de notre recherche, des modèles originaux de structures aléatoires qui prennent en compte un certain nombre de paramètres des structures biologiques réelles. Nous avons appliqué ces modèles dans deux situations. Dans une première phase, nous avons montré que ces structures aléatoires constituent une source de bruit pertinente pour tester la sensibilité et la spécificité des algorithmes de comparaison de structures d'ARN. Dans une deuxième phase, nous avons défini et calculé empiriquement plusieurs Z-valeurs, basées sur les différents modèles aléatoires que nous avons développés, pour RNADISTANCE. Nous avons ainsi pu montrer que la Z-valeur, calculée à partir de l'un de nos modèles de structures aléatoires, est plus efficace que le score brut pour la classification par famille des ARN de petite taille. La Z-valeur calculée à partir de structures aléatoires basées sur le modèle de Markov, permet quant à elle d'améliorer la classification des ARN de grande taille sur le critère du score de comparaison.

Dans une perspective d'élargissement de notre recherche, il serait intéressant de considérer les autres méthodes de comparaison de structures, afin de déterminer si le Z-score se comporte de la même manière. L'étude de la distribution de la Z-valeur pour les différents logiciels pourrait également nous permettre d'établir des seuils de détection lorsque l'on cherche à annoter des ARN par similarité de structures.

La deuxième partie de notre recherche concernait l'étude des algorithmes exacts de prédiction de structure secondaire avec pseudonœuds. Nous avons caractérisé d'une façon différente et complémentaire les classes définies par Condon *et al* [25] par des graphes de cohérence. Nous avons ensuite évalué le compromis entre la complexité en temps et la cardinalité des espaces de prédiction de ces classes. Dans ce but, nous avons codé les structures par des mots et réalisé des bijections vers des langages décrits par des grammaires non ambiguës et non contextuelles. Nous avons également établi une bijection entre les structures de la classes *L&P* et des cartes planaires. Après avoir introduit la classe des pseudonœuds indifférenciés, nous avons établi une bijection entre cette classe et les arbres ternaires. Ces deux dernières bijections nous ont permis d'obtenir les formules closes pour le dénombrement de ces classes. Nous avons ainsi pu démontrer que le coût en termes de complexité en temps de certaines méthodes n'était pas justifié par la taille de leur

espace de prédiction. En effet, les espaces de prédiction de certains de ces algorithmes apparaissent comme étant surdimensionnés tandis que d'autres sont sous-dimensionnés. Par exemple, l'espace de prédiction de *L&P* n'est guère plus grand que celui des structures sans pseudonœud. En revanche, l'espace de prédiction de la classe *R&E*, qui couvre presque l'ensemble des structures existantes, contient également un grand nombre de structures d'une grande complexité, qui ne présentent pas de réalisme biologique, mais dont la prédiction a un coût élevé en termes de complexité en temps. On peut également citer l'algorithme de Cao et Chen, qui possède une forte complexité, alors qu'il ne prédit pas un grand nombre de structures. Cependant, la méthode de calcul de l'énergie qu'il utilise est l'une des plus abouties. L'obtention de grammaires non ambiguës et non contextuelles nous permet d'engendrer aléatoirement des mots de langages qui sont en bijection avec les structures avec pseudonœuds. Nous avons montré que nous pouvions implanter cette bijection des langages non contextuels vers les structures pseudonouées grâce à des algorithmes en temps linéaires et ainsi réaliser cette génération aléatoire de façon efficace.

Ces observations nous conduisent à considérer autrement les algorithmes de prédiction de structure. On peut notamment séparer la notion d'espace de prédiction du calcul et la fonction d'énergie. On peut alors représenter l'espace de recherche des structures par un hypergraphe comme le montrent les travaux de Filkenstein et Roytberg [40] d'une part, et calculer d'autre part les grandeurs qui nous intéressent à partir de ce même hypergraphe. Avec Yann Ponty et Balaji Raman du LIX, nous élaborons une méthode unifiée pour calculer la fonction de partition, les moments de la fonction d'énergie, les probabilités d'appariements, etc ... pour la prédiction de structure secondaire d'ARN avec pseudonœuds.

On peut également s'intéresser au problème du surdimensionnement des espaces de prédiction. Les informations présentes dans les bases de données biologiques indiquent que les pseudonœuds réels sont essentiellement soit de type H, soit simples (une seule structure biologique est répertoriée comme comportant des pseudonœuds simples qui ne soient pas de type H [85]). Les autres pseudonœuds de type *R&E* semblent être constitués essentiellement d'épingles à cheveux embrassées, ou bien il s'agit de pseudonœuds non planaires dont le graphe de cohérence est une clique. Il serait, par conséquent, fructueux de caractériser exactement les structures existantes dans ces bases afin de construire un algorithme dont l'espace de prédiction ciblerait plus précisément ces structures. La réduction de l'espace de prédiction nous permettrait non seulement de réduire la complexité en temps des méthodes existantes, mais aussi de réduire le temps de calcul effectif de ces prédictions.

Ce travail sur les pseudonœuds peut également être appliqué à l'étude des duplex d'ARN. En effet, un duplex d'ARN peut être considéré comme l'appariement d'un ARN avec une boucle d'une structure secondaire appartenant à un autre ARN (voir figure 6.20). La prédiction et la compréhension de ces duplex nous permettrait de mieux comprendre les interactions ARN/ARN.

Nous savons que les classes de *R&E* et *G&D* (la restriction planaire de *R&E*) n'ont pas encore été dénombrées. Pour dénombrer les autres classes, nous avons utilisé des grammaires algébriques non ambiguës. Nous disposons d'une grammaire

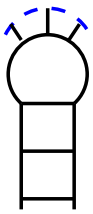


FIGURE 6.20 – ***Représentation d'un duplex d'ARN entre une structure secondaire en noir et un brin libre d'un autre ARN en pointillés bleus.***

ambiguë pour décrire les structures de la classes *R&E* [90] et nous savons construire une grammaire ambiguë pour décrire les structures de la classe *G&D*. Or il n'existe pas de moyens connus pour lever l'ambiguité d'une grammaire algébrique, le problème étant indécidable. Il pourrait être productif d'utiliser une autre méthode de dénombrement pour ces deux classes. Nous savons également que le langage général des structures d'ARN n'est pas algébrique. On peut dès lors se poser la question de la détermination d'un critère ou d'une contrainte sur ces structures qui nous ferait passer d'un système algébrique descriptible par une grammaire non contextuelle à un système non algébrique.

Considérant les résultats de notre étude sur la relation entre complexité et espaces de prédiction, une des questions théoriques intéressantes en découlant pourrait être la détermination d'un critère précis, basé sur l'espace de prédiction des algorithmes, qui nous permettrait de déterminer si la prédiction de cet espace de conformation est un problème NP-Complet ou non. En d'autre termes, on peut se demander quel est l'espace des structures prédictibles en temps polynomial.

Nous n'avons envisagé dans ce travail que les algorithmes exacts de prédiction de structure secondaire d'ARN. Il existe cependant d'importants travaux heuristiques sur les structures tertiaires d'ARN. On peut notamment citer Parisien et Major qui ont développé MC-Fold [81], une heuristique *ab initio* permettant de calculer la structure tertiaire avec pseudonœuds d'un ARN . Leur approche ne permet cependant pas de déterminer la classe de pseudonœuds qu'ils prédisent. Dans cette perspective, nous pourrions développer un algorithme de prédiction exact de structure tertiaire pour lequel nous maîtriserions l'espace de prédiction. Ce type d'approche nous permettrait d'envisager des critères thermodynamiques pour le repliement des séquences d'ARN en structure tertiaire.

Bibliographie

[1] J.P. Abrahams, M. van den Berg, E. van Batenburg, and C. Pleij. Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Nucleic Acid Research*, 18 :3035–3044, 1990.

[2] T. Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104 :45–62, 2000.

[3] J. Allali, Y. d'Aubenton Carafa, C. Chauve, A. Denise, C. Drevet, P. Ferraro, D. Gautheret, C. Herrbach, F. Leclerc, A. de Monte, A. Ouangraouda, M.F. Sagot, C. Saule, M. Termier, C. Thermes, and H. Touzet. Benchmarking RNA secondary structure comparison algorithms. *Actes des Journées Ouvertes de biologie, Informatique et Mathématiques, Lille*, pages 67–68, 2008.

[4] J. Allali and M.F. Sagot. A multiple layer model. to compare RNA secondary structures. *Software : Practice and Experience*, 2007.

[5] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology.*, 215 :403–410, 1990.

[6] E.S. Andersen, M.A. Rosenblad, N. Larsen, J.C. Westergaard, J. Burks, I.K. Wower, J. Wower, J. Gorodkin, T. Samuelsson, and C. Zwieb. The tmRDB and SRPDB resources. *Nucleic Acids Res*, 34 :163–168, 2006.

[7] M.S. Andronescu, C. Pop, and A.E. Codon. Improved free energy parameters for RNA pseudoknotted secondary structure prediction. *RNA*, 10 :26–42, 2010.

[8] J.P. Bachellerie, J. Cavaillé, and A. Hüttenhofer. The expanding snoRNA world. *Biochimie*, 84(8) :775–790, 2002.

[9] J.K. Baker. Trainable grammar for speech recognition. *Speech Communication Paper for the 97th Meeting of the Acoustical Society of America.*, pages 547–550, 1979.

[10] F.H.D. Van Batenburg, A.P. Gultyaev, and C.W.A. Pleij. An apl-programmed genetic algorithm for the prediction of RNA secondary structure. *Journal of Theoretical Biology*, 174 :269–280, 1995.

[11] G. Blin, A. Denise, S. Dulucq, C. Herrbach, and H. Touzet. Alignments of RNA structures. *IEEE/ACM Transactions on Computational Biology and Bioinformatics.*, 7(2) :309–322, 2010.

[12] G. Blin and H. Touzet. How to compare arc annotated sequences : The alignment hierarchy. *SPIRE.*, 4209 of LNCS, 2006.

[13] A.F. Bompfünewerer, C. Flamm, C. Fried, G. Fritzsch, I. L. Hofacker, J. Lehmann, K. Missal, A. Mosig, B. Müller, S.J. Prohaska, B.M.R. Stadler, P.F. Stadler, A. Tanzer, S. Washietl, and C. Witwer. Evolutionary patterns of non-coding RNAs. *Theory in Biosciences*, 123(4) :301 – 369, 2005.

[14] J.W. Brown. The ribonuclease P database. *Nucleic Acids Res*, 27 :314, 1999.

[15] J.J. Cannone, S. Subramanian, M.N. Schnare, J.R. Collett, L.M. D'Souza, Y. Du, B. Feng, N. Lin, L.V. Madabusi, K.M. MÜller, N. Pande, Z. Shang, N. Yu, and R.R. Gutell. The comparative RNA web (crw) site : An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3 :2, 2002.

[16] S. Cao and S-J. Chen. Predicting RNA folding thermodynamics with a reduced chain representation model. *RNA*, 11 :1884–1897, 2005.

[17] S. Cao and S-J. Chen. Predicting RNA pseudoknot folding thermodynamics. *Nucleic Acid Research*, 34 :2634–2652, 2006.

[18] S. Cao and S-J Chen. Predicting structures and stabilities for h-type pseudoknots with interhelix loops. *RNA*, 15 :696–706, 2009.

[19] T.R. Cech. Self-splicing of group I introns. *Annual Review of Biochemistry*, 59 :543–568, 1990.

[20] X. Chen, S-M. He, D. Bu, F. Zhang, Z. Wang, R. Chen, and W. Gao. Flexstem : improving predictions of RNA secondary structures with pseudoknots by reducing the search space. *Bioinformatics*, 24 :1994–2001, 2008.

[21] P. Clote. Combinatoric of satured secondary structures of RNA. *Journal of Computational Biology*, 13(9) :1640–1657, 2006.

[22] P. Clote, F. Ferré, E. Kranakis, and D.Krizanc D. Structural RNA has lower folding energy than random RNA of the same dinucleotide frequency. *RNA*, 11(5) :578–591, 2005.

[23] P. Clote, E. Kranakisc, D. Krizancd, and L. Stach. Asymptotic expected number of base pairs in optimal secondary structure for random RNA using the Nussinov–Jacobson energy model. *Discrete Applied Mathematics*, 155(6-7) :759–787, 2006.

[24] J.P. Comet, J.C. Aude, E. Glémet, J.L. Risler, A. Hénaut, P.P. Slonimski, and J.J. Codani. Significance of Z-score statistics of Smith-Waterman scores for protein alignments. *Computers & Chemistry*, 23 :317–331, 1999.

[25] A. Condon, B. Davy, B. Rastegari, S. Zhao, and F. Tarrant. Classifying RNA pseuknotted structures. *Theorical computer science*, 320 :35–50, 2004.

[26] ENCODE Project Consortium. Identification and analysis of functional elements in 1% of the human genome by the encode pilot project. *Nature*, 459 :927–930, 2007.

[27] L. Dai, N. Toor, R. Olson, A. Keeping, and S. Zimmerly. Database for mobile group II introns. *Nucleic Acids Research*, 31 :424–426, 2003.

[28] E.T. Dam, K. Pleij, and D. Draper. Structural and functional aspects of RNA pseudoknots. *Biochemistry*, 31(47) :11665–11676, 1992.

[29] K. Darty, A. Denise, and Y. Ponty. Varna : Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15) :1974–1975, 2009.

[30] A. Denise, Y. Ponty, and M. Termier. Controlled non-uniform random generation of decomposable structures. *Theor. Comput. Sci.*, 411(40-42) :3527–3552, 2010.

[31] A. Denise, O. Roques, and M. Termier. Random generation of words of context-free languages according to the frequencies of letters. *Trends in Mathematics*, page 113–125, 2000.

[32] J. Deogun, E. Donis, O. Komina, and F. Ma. RNA secondary structure prediction with simple pseudoknots. *In Proc Second Asia-Pacific Bioinformatics Conference 2004*, pages 239–246, 2004.

[33] R.M. Dirks and N.A. Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J Comput Chem*, 24 :1664–1677, 2003.

[34] R. Donaghey and W.L. Shapiro. Motzkin numbers. *Journal of Combinatorial theory*, 23 :291–301, 1976.

[35] P. Duchon, P. Flajolet, G. Louchard, and G.Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability, and Computing*, 13(4-5) :577–625, 2004. Special issue on Analysis of Algorithms.

[36] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, probabability and computing*, 13(4-5) :577–625, 2004.

[37] R. Durbin, S.R. Eddy, A. Krogh, and G.J. Mitchinson. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[38] P. Ferraro and C. Godin. An edit distance between quotiented trees. *Algorithmica.*, 36 :1–39, 2003.

[39] A.P. van Gultyaev F.H.D. Batenburg, C.W.A. Pleij, J. Ng, and J. Oliehoek. Pseudobase : a database with RNA pseudoknots. *Nucleic Acid Research*, 28 :201–204, 2000.

[40] A.V. Finkelstein and M.A. Roytberg. Computational of biopolymers : A general approach to different problems. *BioSystems*, 30 :1–19, 1993.

[41] A. Fire, S. Xu, M.K. Montgomery, S.A. Kostas, S.E. Driver, and C.C. Mello. Potent and specific genetic interference by double-stranded RNA in *caenorhabditis elegans*. *Nature*, 391 :806–811, 1998.

[42] P. Flajolet and R. Sedgewick. *Analytic Combinatorics.* Cambridge University Press, 2009.

[43] P. Flajolet and P. Van Cutsem Zimmermann. A calculus for the random generation of labelled combinatorial structures. *Theorical computer sciences*, 132 :1–35, 1994.

[44] S. Gao and K. Ding. A graphical criterion of planarity for RNA secondary structures with pseudoknots in rivas–eddy class. *Theorical computer science*, 395 :47–56, 2008.

[45] C. Gaspin, J. Cavaille, G. Erauso, and J.P. Bachellerie. Archaeal homologs of eukaryotic methylation guide small nucleolar RNAs : lessons from the pyrococcus genomes. *Journal of Molecular Biology*, 297(4) :895–906, 2000.

[46] J. Gevertz, H.H. Gan, and T. Schlick. In vitro RNA random pools are not structurally diverse : A computational analysis. *RNA*, 11 :853–863, 2005.

[47] M. Ghildiyal and P.D. Zamore. Small silencing RNAs : an expanding universe. *Nature Review Genetics*, 10(2) :94–108, 2009.

[48] R. Giegerich, B. Voß, and M. Rehmsmeier. Abstract shapes of RNA. *Nucleic Acids Research.*, 32(16) :4843–4851, 2004.

[49] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy. Rfam : an RNA family database. *Nucleic Acids Research*, 31(1) :439–441, 2003.

[50] V. Guignon, C. Chauve, and S. Hamel. An edit distance between RNA stem-loops. *Lecture Notes in Computer Science.*, 3772 :335–347, 2005.

[51] A.P. Gultayev, F.H.D. Van Batenburg, and C.W.A. Pleij. The computer simulation of RNA folding pathways using a genetic algorithm. *Journal of molecular biology*, 250 :37–51, 1995.

[52] B. Han, B. Dost, V. Bafna, and S. Zhang. Structural alignment of pseudoknotted RNA. *Journal of Computational Biology*, 15(5) :489–504, 2008.

[53] C. Haslinger and P. Stadler. RNA structures with pseudo-knots : Graph-theorical, combinatorial, and statistical properties. *Bulletin of Mathematical Biology*, 61 :437–467, 1999.

[54] P. Haugen, D. M. Simon, and D. Bhattacharya. The natural history of group I introns. *Trends Genetics.*, 21(2) :111–119, 2005.

[55] M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz. Local similarity in RNA secondary structures. *Proceedings of the IEEE Bioinformatics Conference.*, 2 :159–168, 2003.

[56] C. Herrbach. Etude algorithmique et statistique de la comparaison de structures secondaires d'ARN. *PhD Thesis University Bordeau I.*, 2007.

[57] I. L. Hofacker, P. Schuster, and P. F. Stadler. Combinatorics of RNA secondary structures. *Discr. Appl. Math*, 89 :207–237, 1996.

[58] I.L. Hofacker, W. Fontana, P.F. Stadler, S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte f. Chemie.*, 1994.

[59] H. Jabbari, A. Condon, A. Pop, C. Pop, and Y. Zhao. Hfold : RNA pseudoknotted secondary structure prediction using hierarchical folding. *7th Workshop on Algorithms in Bioinformatics (WABI), Philadelphia, USA. Algorithms in Bioinformatics, Lecture Notes in Computer Science (LNCS)*, 4645 :323–334, 2007.

[60] T. Jiang, L. Wang, and K. Zhang. Alignment of trees - an alternative to tree edit. *Theorical Computer Science.*, 143 :137–148, 1995.

[61] E. Y. Jin, , and C.M. Reidys. RNA pseudoknot structures with arc-length $\geq$ 3 and stack-length $\geq \sigma$. *Discrete Appl. Math.*, 158(1) :25–36, 2010.

[62] E.Y. Jin, J. Qin, and C.M. Reidys. Combinatorics of RNA structures with pseudoknots. *Bull Math Biol.*, 70(1) :45–67, 2008.

[63] E.Y. Jin and C.M. Reidys. Asymptotic enumeration of RNA structures with pseudoknots. *Bull Math Biol.*, 70(4) :951–970, 2008.

[64] S. Karlin and S.F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general schemes. *Procedings of the National Academic of Science, USA*, 87 :2264–2268, 1990.

[65] R.J. Klein, Z. Misulovin, and S.R. Eddy. Noncoding RNA genes identified in AT-rich hyperthermophiles. *Procedings of the National Academy of Sciences of U.S.A.*, 99(11) :7542–7547, 2002.

[66] K. Lari and S.J. Young. The estimation of stochastic context-free grammars using the "inside-outside" algorithm. *Computer Speech and Language.*, 4 :35–56, 1990.

[67] N. Larsen and C. Zwieb. SRP-RNA sequence alignment and secondary structure. *Nucleic Acid Research*, 19(2) :209–215, 1990.

[68] D.J. Lipman, W.J. Wilbur, T.F. Smith, and M.S. Waterman. On the statistical significance of nucleic acid similarities. *Nucleic Acid Research*, 12(1) :215–226, 1984.

[69] L.M.M.Soares and J. Valcarcel. The expanding transcriptome : the genome as the 'book of sand'. *The EMBO Journal*, 25 :923–931, 2006.

[70] W.A. Lorenz, Y. Ponty, and P. Clote. Asymptotics of RNA shapes. *Journal of Computational Biology*, 15(1) :31–63, Jan-Feb 2008.

[71] R. B. Lyngsø and C. N. Pedersen. RNA pseudoknot prediction in energy based models. *Journal of computational biology*, 7 :409–428, 2000.

[72] R.B. Lyngsø and C.N.S. Pedersen. Pseudoknots in RNA secondary structures. *Proceedings of* 4^{th} *Annual Internationnal Conference on Computational Molecular Biology(RECOMB), ACM Press*, pages 201–209, 2000.

[73] C. Marck and H. Grosjean. tRNomics : Analysis of tRNA genes from 50 genomes of eukarya, archaea, and bacteria reveals anticodon-sparing strategies and domain-specific features. *RNA*, 8 :1189–1232, 2002.

[74] D.H. Mathews, J. Sabina, M. Zuker, and D.H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288 :911–940, 1999.

[75] J.S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymer*, 29 :1105–1119, 1990.

[76] M. Möhl, S. Will, and R. Backofen. Lifting prediction to alignment of RNA pseudoknots. *Journal of Computational Biology*, 17 :429–442, 2010.

[77] M. E. Nebel. Combinatorial properties of RNA secondary structures. *Journal of Computational Biology*, 9(3) :541–574, 2003.

[78] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3) :443–453, 1970.

[79] R. Nussinov, G. Pieczenik, J.R. Griggs, and D.J. Kleitman. Algorithms for loop matching. *SIAM J. Appl. Math.*, 35 :68–82, 1978.

[80] A. Ouangraoua, P. Ferraro, L. Tichit, and S. Dulucq. Local similarity between quotiented ordered trees. *Journal of Discrete Algorithms.*, 2007.

[81] M. Parisien and F. Major. The MC-Fold and MC-sym pipeline infers RNA structure from sequence data. *Nature*, 452 :51–55, 2008.

[82] C.W.A. Pleij and L. Bosch. RNA processing, part a : General methods. *Methods in Enzymology*, 180 :289–303, 1989.

[83] Y. Ponty. Modsation de sences gmiques structur, gration aloire et applications. *PhD Thesis University Paris-Sud XI.*, 2006.

[84] Y. Ponty, M. Termier, and A.Denise. GenRGenS : Software for generating random genomic sequences and structures. *Bioinformatics*, 22(12) :1534–1535, 2006.

[85] B. Rastegari and A. Condon. Parsing nucleic acid pseudoknotted secondary structure : algorithm and applications. *Journal of Computational Biology*, 14 :16–32, 2007.

[86] R.B. Lyngsø R.B., M. Zucker, and C.N.S. Pedersen. Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics*, 15 :440–445, 1999.

[87] J. Reeder and R. Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5 :104, 2004.

[88] J. Ren, B. Rastegari, A. Condon, and H.H. Hoos. Hotknots : Heuristic prediction of RNA secondary structures including pseudoknots. *RNA*, 11 :1494–1504, 2005.

[89] E. Rivas and S.R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of molecular biology*, 285 :2053–2068, 1999.

[90] E. Rivas and S.R. Eddy. The language of RNA : a formal grammar that includes pseudoknots. *Bioinformatics*, 16 :334–340, 2000.

[91] E. Rivas and S.R. Eddy. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, 16(7) :583–605, 2000.

[92] E. A. Rødland. Pseudoknots in RNA secondary structures : representation, enumeration, and prevalence. *Journal of Computational Biology*, 13(6) :1197–1213, 2006.

[93] J. Ruan, G.D. Stormo, and W. Zhang. An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics*, 20 :58–66, 2004.

[94] R. Saldhana, G. Mohr, M. Belfort, and A.M. Lambowitz. Group I and group II introns. *The FASEB Journal*, 7 :15–24, 1993.

[95] C. Saule, M. Régnier, J-M. Steyaert, and A. Denise. Counting RNA pseudoknotted structures. *Journal of Computational Biology.*, to appear.

[96] C. Saule, C. Wallon, and A. Denise. Uniform and non-uniform random generation of RNA secondary structures with pseudoknots (extended abstract). *in proceedings of GASCOM'10*, 2010.

[97] N.J.A. Sloane and S. Plouffe. *The Encyclopedia of Integer Sequences.* Academic Press, 1995.

[98] S. Smit, K. Rother, J. Heringa, , and R. Knight. From knotted to nested RNA structures : a variety of computational methods for pseudoknot removal. *RNA*, 14(3) :410–416, 2008.

[99] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147 :195–197, 1981.

[100] J. Sperschneider and A. Datta. Knotseeker : Heuristic pseudoknot detection in long RNA sequences. *RNA*, 14 :630–640, 2008.

[101] D.W. Staple and S.E. Butcher. Pseudoknots : RNA structures with diverse functions. *PLoS Biology*, 3(6) :956–959, 2005.

[102] P.R. Stein and M.S. Waterman. On some sequences generalising the Catalan and Motzkin numbers. *Discrete Mathematics*, 26 :261–272, 1978.

[103] S. Steinberg, F. Leclerc, and R. Cedergren. Structural rules and conformational compensations in the tRNA l-form1. *Journal of Molecular Biology*, 266(2) :269–282, 1997.

[104] I. Tinoco and C. Bustamente. How RNA folds. *Journal of Molecular Biology*, 293 :271–281, 1999.

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

[106] Y. Uemura, A. Hasegawa, S. Kobayashi, and T. Yokomori. Tree adjoining grammars for RNA structures prediction. *Theorical Computer Science*, 210 :277–303, 1999.

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

[108] G. Vernizzi, H. Orland, and A. Zee. Enumeration of RNA structures by matrix models. *Phys. Rev. Lett.*, 94 :168103, 2005.

[109] T. R. S. Walsh and A.B. Lehman. Counting rooted maps by genus. iii : Nonseparable maps. *J. Combinatorial Theory Ser. B*, 18 :222–259, 1975.

[110] M. S. Waterman. Secondary structure of single-stranded nucleic acids. *Advances in Mathematics Supplementary Studies*, 1(1) :167–212, 1978.

[111] F. Weinberg and M.E. Nebel. Extending stochastic context-free grammars for an application in bioinformatics. *proceedings of the 4th International Conference on Language and Automata Theory and Applications (LATA2010).*, 6031 :585–595, 2010.

[112] S.A. Woodson. Structure and assembly of group I introns. *Current Opinion in Structural Biology*, 15(3) :324 – 330, 2005.

[113] A. Xayaphoummine, T. Bucher, and A. Isambert. Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots. *Nucleic Acid Research*, 33 :605–610, 2005.

[114] A. Xayaphoummine, T. Bucher, F. Thalmann, and H. Isambert. Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations. *PNAS*, 100 :15310–15315, 2003.

[115] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18 :1245–1262, 1989.

[116] M. Zucker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acid Research*, 9 :133–148, 1981.

[117] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science.*, 244 :48–52, 1989.

[118] M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acid Research*, 31 :3406–34015, 2003.

Table des figures

Liste des tableaux

Annexe A

Courbes ROC

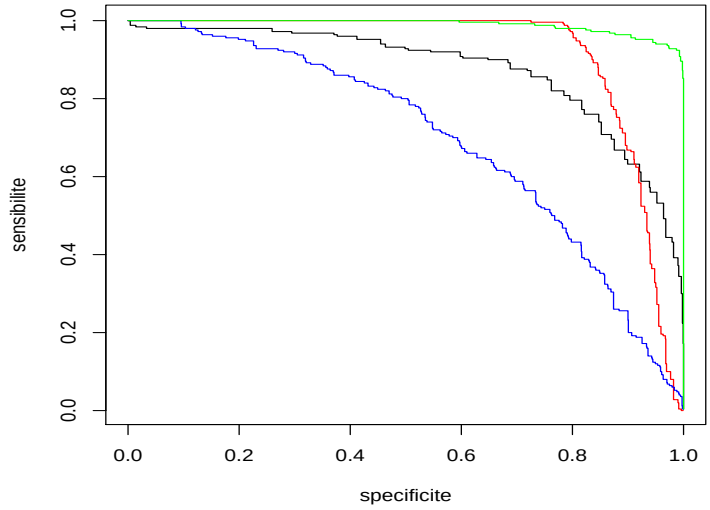


FIGURE A.1 – ***Courbes ROC obtenues à partir des structures de miRNA (Noire : Brut, Rouge : Markov ordre 1 , Bleue : Topologie fixe, Verte : Topologie nombre de tiges fixe).***

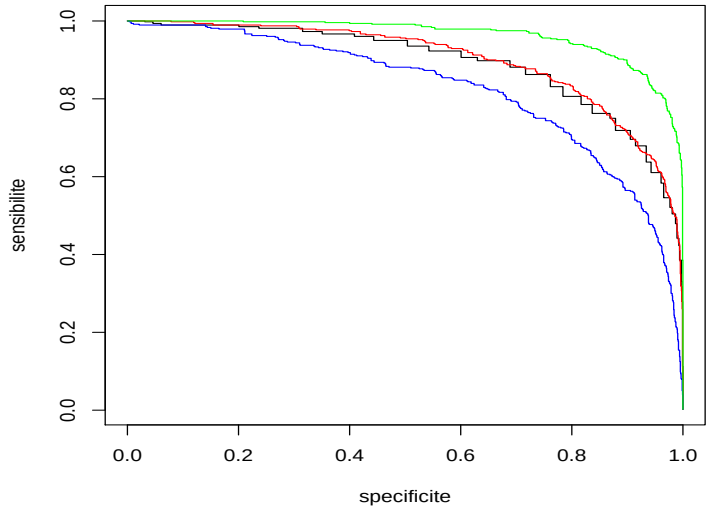


FIGURE A.2 – ***Courbes ROC obtenues à partir des structures de tRNA repliées par Mfold (Noire : Brut, Rouge : Markov ordre 1 , Bleue : Topologie fixe, Verte : Topologie nombre de tiges fixe).***

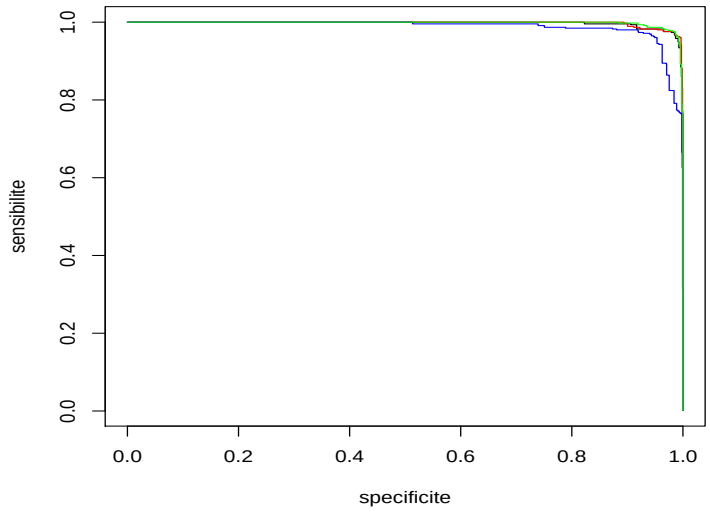


FIGURE A.3 – ***Courbes ROC obtenues à partir des structures de tRNA issues de la base de données Gutell (Noire : Brut, Rouge : Markov ordre 1 , Bleue : Topologie fixe, Verte : Topologie nombre de tiges fixe).***

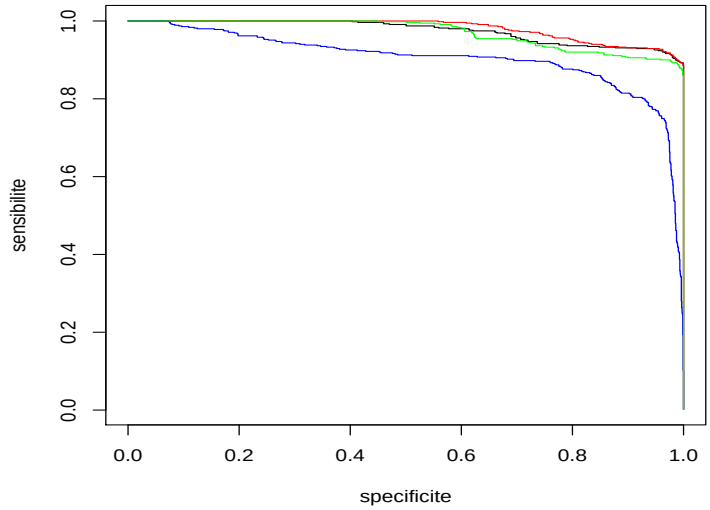


FIGURE A.4 – ***Courbes ROC obtenues à partir des structures de SRP de plantes (Noire : Brut, Rouge : Markov ordre 1 , Bleue : Topologie fixe, Verte : Topologie nombre de tiges fixe).***

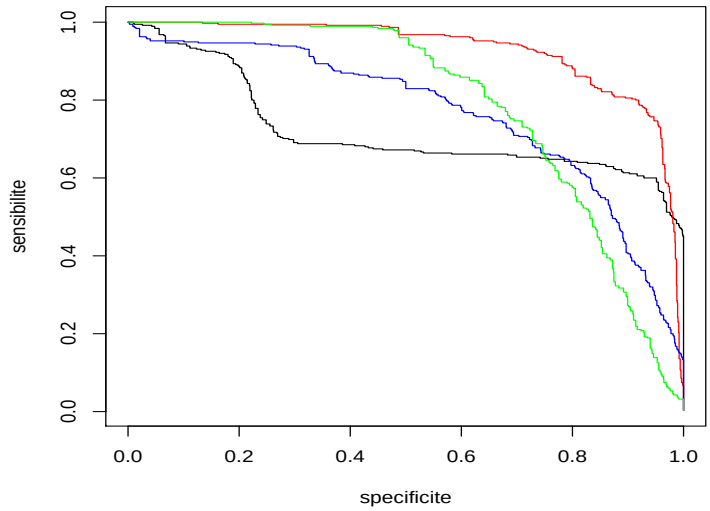


FIGURE A.5 – ***Courbes ROC obtenues à partir des structures de SRP de fungi (Noire : Brut, Rouge : Markov ordre 1 , Bleue : Topologie fixe, Verte : Topologie nombre de tiges fixe).***