Modélisation de séquences génomiques structurées, génération aléatoire et applications

Yann Ponty

N° d'ordre : 8480

THÈSE

présentée à l'université Paris-Sud à Orsay

pour l'obtention du grade de Docteur en Sciences

Spécialité informatique

—

Modélisation de séquences génomiques structurées, génération aléatoire et applications

Yann PONTY

—

Soutenue le 29 Novembre 2006 devant le jury composé de :

Mr Joffroy BEAUQUIER	*Président*
Mr Alain DENISE	*Directeur*
Mr Philippe FLAJOLET	*Rapporteur*
Mr Daniel GAUTHERET	*Examinateur*
Mr Markus NEBEL	*Examinateur*
Mr Jacques NICOLAS	*Examinateur*
Mr Eric RIVALS	*Rapporteur*

A mes parents,
pour m'avoir rendu bien incapable
d'énumérer ce qu'ils m'ont offert.

TABLE DES MATIÈRES

RÉSUMÉ

En bioinformatique, de nombreux modèles s'attachent à formaliser un lien entre une propriété observée dans une entité génomique et son impact sur le rôle, la fonction de celle ci. Cependant, certaines de ces propriétés peuvent n'être que le produit "émergeant" d'un ensemble de phénomènes déjà étudiés et compris, ou encore n'être que le résultat du hasard, du bruit de fond. La distinction entre les phénomènes nouveaux, qui méritent l'attention des biologistes et bioinformaticiens, et ceux artefactuels, qui sont déjà expliqués par ce qui est connu, fait appel au concept statistique de score de significativité. Ce dernier se base sur la probabilité d'apparition d'un phénomène dans une population présentant un certain nombre de caractéristiques. Dans le cas des séquences et structures des macromolécules (ADN, ARN ou protéines), cette probabilité peut être évaluée soit mathématiquement, soit par le biais d'une génération aléatoire. Des séquences sont alors engendrées selon des contraintes associées aux phénomènes déjà compris, et le score de significativité est évalué expérimentalement sur cet échantillon. La principale difficulté liée à cette approche réside dans la mise sous contrainte des séquences engendrées. Comment peut-on à la fois exprimer ces contraintes tout en gardant un certain contrôle sur la distribution des séquences, le tout en consacrant le moins de temps possible à la génération ?

La première contribution de cette thèse présente un premier élément de réponse à ce problème. A travers le développement et la diffusion de `GenRGenS`, un logiciel dédié à la génération aléatoire de séquences et structures génomiques, nous avons fourni à la communauté un ensemble d'algorithmes de génération aléatoire pour différentes classes de modèles de séquences. Parmi celles-ci, le formalisme des grammaires pondérées, étendu par Denise *et al.* [29], qui adjoint des pondérations aux symboles terminaux d'une grammaire non-contextuelle, ce qui induit une distribution de probabilités sur les séquences. Je propose un mécanisme de construction d'une grammaire pondérée réalisant le produit d'une grammaire non-contextuelle et d'un modèle de Markov. J'ai par ailleurs obtenu un algorithme de génération en taille approchée linéaire pour ce formalisme par une adaptation de la mé-tho-de dite de Boltzmann, introduite par Duchon *et al.* [36] pour la génération aléatoire d'objets combinatoires. Enfin, la distribution de probabilité induite par le système de pondérations peut être utilisée pour contraindre les fréquences d'apparition des symboles terminaux dans les séquences engendrées. Je me suis dans un premier temps intéressé à l'évaluation des fréquences d'apparition des non-terminaux pour une pondération donnée. Il en est ressorti deux types d'approches : Une première basée sur le calcul symbolique fournit la limite des fréquences quand la taille des séquences est grande, et une deuxième inspirée de la programmation dynamique. C'est cette deuxième méthode qui a rendu possible le développement du logiciel `GrGFreqs`, qui calcule la pondération réalisant des fréquences attendues par une approche optimisation.

Ces travaux ont fait l'objet d'une application à l'ARN et plus particulièrement à sa structure secondaire, qui consiste en une restriction planaire de l'ensemble de ses liaisons. L'utilisation de symboles terminaux disjoints au sein des différents types de sous-structures (hélices, boucles, renflements ...) a permis, via l'adjonction de pondérations, de contraindre en moyenne les tailles et nombres d'occurrences de ces sous-structures au sein des structures engendrées. L'utilisation du logiciel `GrGFreqs` implémentant l'approche optimisation a alors permis le calcul de jeux de pondérations réalisant les valeurs de ces paramètres pour quelques grandes familles d'ARN. D'autre part, le calcul de ces valeurs pour des structures d'ARN issues de la cristallographie m'a amené à m'intéresser au problème de la planarisation de structure d'ARN. Par planarisation, on entend l'extraction dans un ensemble de liaisons d'un sous-ensemble planaire, et maximal selon un certain critère. Je me suis donc intéressé à quelques familles de critères d'optimalité. Je propose un algorithme en $O(n^2+nm)$, où n est le nombre de bases et m le nombre de liaisons, pour ce problème dans l'hypothèse d'un critère d'optimalité additif. Il s'agit à notre connaissance de la première solution non heuristique à ce problème.

Mais la génération aléatoire de séquences connaît d'autres champs d'applications que ceux sus-cités. En particulier, celle-ci peut être utilisée pour évaluer la sensibilité d'algorithmes heuristiques, algorithmes qui sacrifient un peu de leur capacité de prédiction au profit d'une complexité optimisée. Je me suis intéressé à la génération aléatoire de chemins culminants, une variété de chemins dans le quart de plan qui modélisent particulièrement bien le cheminement des scores dans l'heuristique XDrop centrale à l'algorithme BLAST. Ces chemins sont composés de pas $(+1,+a)$ et $(+1,-b)$ et relient $(0,0)$ à (n,k), où k est la plus grande ordonnée jamais atteinte au cours de la marche. J'ai prouvé que le langage défini par ces chemins n'est pas algébrique, c'est à dire qu'il n'existe pas de grammaire non-contextuelle les décrivant. Dans le cadre d'une collaboration avec L. Noe et G.Kucherov, j'ai ensuite découvert une décomposition de ces chemins qui a permis l'élaboration d'un algorithme de génération aléatoire uniforme de complexité linéaire après un précalcul en $O(n^3)$. Cette décomposition a été utilisée pour calculer la probabilité d'apparition d'un motif appelé graine (seed) dans le chemin, probabilité équivalente à la sensibilité de cette graine dans l'algorithme FLASH. Nous avons ainsi mis en évidence un biais dans les précédentes méthodes d'évaluation. J'ai poursuivi cette étude sur un plan combinatoire, et ai obtenu, dans une collaboration avec M. Bousquet-Mélou, des algorithmes de génération aléatoire linéaires basés sur des méthodes par rejet dans les trois cas $a<b$, $a=b$ et $a>b$. La preuve de cette linéarité nous a amenés à pousser plus en avant l'étude de ces chemins d'un point de vue énumératif, et a fait apparaître des régimes différents dans ces trois cas, ainsi que des séries génératrices de dénombrement transcendantes.

1. REMERCIEMENTS

Il m'aurait été impensable de ne pas débuter cette liste, non exhaustive, par un remerciement adressé à Alain DENISE, mon directeur de thèse et *collaborateur* depuis maintenant 5 ans. N'ayant réfréné aucun de mes enthousiasmes mais ayant su m'insuffler une énergie nouvelle au creux de la vague, ayant favorisé mon intégration dans la communauté tout en m'ayant incité à consacrer le temps nécessaire à mes enseignements, ayant insisté sur la nécessité de découvrir les règles implicites qui régissent nos communautés scientifiques tout en m'ayant prémuni contre un cynisme précoce, Alain est le directeur que je souhaite à tout doctorant en bioinformatique. Sur le plan scientifique, sa rigueur méthodologique combinée à une volonté d'application aux problématiques biologiques sont une source d'inspiration permanente.

Je souhaite ensuite adresser un remerciement à Philippe FLAJOLET et Eric RIVALS, qui ont eu la gentillesse d'accepter d'être les rapporteurs du présent manuscrit. Philippe, par l'acuité de ses intuitions et le ton, toujours informel et détendu, qu'il a su insuffler à nos discussions, est l'un de ceux qui ont fait le plus progresser ma réflexion sur la génération aléatoire, et plus généralement sur la combinatoire. Eric a pour sa part fourni un travail de relecture proprement herculéen et impitoyable, et je tiens tout particulièrement à le remercier de sa patience. Je m'estime extrêmement chanceux d'avoir eu pour rapporteur un scientifique d'une culture et d'une ouverture scientifique aussi vastes.

Je remercie Joffroy BEAUQUIER d'avoir accepté la présidence du jury, il a contribué à la construction de ma sensibilité algorithmique au travers de cours d'algorithmique parallèle suivis en licence/maîtrise, ce qui m'a permis maintes fois depuis de briller en société en expliquant les raisons du crash d'ARIANE ou l'existence intrinsèque de scénarios-catastrophe de probabilités non-nulles susceptibles de faire défaillir les tours de contrôle. Bien qu'étant issu d'une communauté scientifique assez éloignée de la bioinformatique, il a su s'intéresser à mes travaux, comme en ont témoigné les questions et remarques très pertinentes qu'il m'a adressées.

Daniel GAUTHERET, Jacques NICOLAS et Markus NEBEL m'ont fait l'honneur de leur présence au sein du jury. Leurs éclairages nouveaux et complémentaires sur mes travaux, exprimés à travers des questions astucieuses et des conseils francs, influenceront pour beaucoup mes travaux de post-thèse.

Une thèse est un exercice de longue haleine, s'effectuant dans un environnement dont dépend pour beaucoup l'impression laissée au doctorant. Dans mon cas, cet environnement s'est avéré

tellement plaisant et stimulant qu'il me vient presque parfois des envies de *rempiler*. Ceci m'amène à remercier toute l'équipe Bioinformatique du LRI : Claire HERRBACH tout d'abord, qui a apparemment pu me supporter 3 ans comme co-bureau, et aura au final *presque* réussi à me faire comprendre son algorithme de programmation dynamique en $\mathcal{O}(n^6)$; Jean Paul FOREST, notre père spirituel à tous, dont la conversation fut toujours riche, profonde, et d'application immédiate (un bioinformaticien, en somme ...). Sa disponibilité et sa volonté d'aider ses jeunes collègues thésards sont pour beaucoup dans la création d'un esprit d'équipe au sein de ce collectif; Romain RIVIERE, dont j'ai pu apprécier la rigueur mathématique et les profondes compétences techniques, sans parler du jonglage; Patrick AMAR, dont la gentillesse sans limite et la constante bonne humeur furent de vrais soutiens dans les heures sombres; Stéphane VIALETTE, dont les intuitions fulgurantes en algorithmique, un peu effrayantes au début, se sont avérées au final extrêmement stimulantes et profondément enrichissantes; Mohamed ELATI dont le calme, la discrétion et la modestie n'arrivent pas à dissimuler un vrai bioinformaticien; Céline ROUVEIROL, dont l'empathie et l'humanité m'ont surpris et marqué; Annelyse THÉVENIN, qui sait tout le peu de mal que je pense d'elle; Cédric SAULE, qui m'a fait des remarques très pertinentes sur le présent manuscrit et avec qui j'espère collaborer dans un futur proche à l'application des présents travaux; Bastien RANCE et Lucie GENTILS, avec qui j'ai partagé beaucoup plus que quelques cafés ou thés; Sarah COHEN-BOULAKIA, qui connaît les rouages et m'aura dénudé d'un peu de ma naïveté universitaire; enfin Christine FROIDEVAUX, notre chef bien-aimée, à qui j'attribue pour beaucoup dans la réussite de cette équipe bioinformatique.

Dans un environnement à peine élargi (de quelques centaines de mètres seulement ...), je voudrais inclure dans ces remerciements les collègues et amis du LIMSI, de l'IGM, de l' IBBMC et du CGM, qui ont tous activement participé à l'émergence d'un pôle bioinformatique à Orsay, entre autre à travers l'action du PPF Bioinfo. Parmi eux, je distinguerai : Michel TERMIER, éternel amateur de bonne discussion, de bonne chaire et de controverses éclairées et éclairantes. Il est l'un des pères de la bioinformatique à Orsay et, à un niveau plus personnel, l'un de ceux qui m'ont montré la voie. Sa persistance dans sa volonté de me voir traiter certains problèmes est à l'origine de nombreuses des contributions de cette thèse; David ABERGEL, qui est l'un des premiers biologistes que j'ai rencontré qui ait *fait la moitié du chemin* en direction de l'informatique. Notre rencontre à un instant où moi-même cherchais à comprendre les problématiques biologiques fut marquante et extrêmement enrichissante; Pierre GROS, binôme de toujours, dont les volutes et les soupirs de médecin-libéral précèdent toujours de peu l'énoncé de concepts novateurs et parfois non dénués d'implications théoriques, quoi qu'il en dise; Jean-Pierre ROUSSET, qui est l'un des orateurs les plus bluffants de clarté qu'il m'ait jamais été donné d'entendre; Julie BERNAUER, qui est un autre exemple de biologiste (voire, pire, de chimiste) tombée amoureuse du codage et de la technologie informatique. N'en doutons pas, ces gens-là domineront le monde, un jour; Yves D'AUBENTON-CARAFA qui a radicalement modifié, au cours de deux courtes discussions, mon point de vue sur mes propres travaux de thèse, m'amenant à une réflexion nécessaire sur les modèles *empilés* de séquences aléatoires, réflexion que j'espère mener au cours de travaux futurs.

Je souhaite en outre remercier mes coauteurs Laurent NOE, et Gregory KUCHEROV et Mireille BOUSQUET-MÉLOU, pour la patience dont ils ont témoigné, confrontés à l'inexpérience d'un très jeune chercheur.

Un grand merci aussi à Philippe DUCHON, Christian CHOFFRUT, l'équipe ALGO de l'INRIA Rocquencourt et plus généralement à toute la communauté ALEA, pour m'avoir offert l'opportunité de présenter mes travaux du moment, présentations ayant données lieu à des échanges féconds. Au sein de cette dernière, je souhaite distinguer : Mireille RÉGNIER, qui veille sur les jeunes chercheurs de cette communauté comme une mère bienveillante; Sylvie CORTEEL, qui hormis Alain, doit être la seule personne à avoir lu intégralement mon mémoire de DEA; Cyril BANDERIER, dont l'érudition et la curiosité auront nourri bien des discussions extrêmement stimulantes au cours des éditions successives de ces journées à Luminy.

La fin de la rédaction du manuscrit ayant quelque peu empiété sur ma période d'entrée en postdoc au sein du Boston College, je remercie Peter CLOTE pour sa compréhension, m'ayant permis de consacrer un temps nécessaire à la maturation du manuscrit, et facilitant mon intégration dans ce nouvel environnement. A ce titre, je tiens aussi à remercier Jérôme WALDISPUHL, Tomasz SZATANEK et Tara MARINI pour avoir rendu extrêmement plaisante mon arrivée à Boston, allant jusqu'à me décharger de certaines contingences matérielles qu'il m'arrivait de négliger dans la période de présoutenance (manger, boire, maintenir un semblant de vie sociale ...). Je leur dois sans doute, parmi tant d'autres choses, d'avoir repoussé ma première dépression nerveuse de quelques années.

Au cours de ma thèse, il m'a été donné la chance d'enseigner dans le cadre d'un monitorat. J'en remercie en tout premier lieu le CIES, les bénéfices de cette expérience dépassant de loin mes attentes, et l'ensemble du personnel enseignant du département d'informatique de l'IUT d'Orsay pour son accueil chaleureux et son souci de considérer les moniteurs comme des enseignants à part entière. Enfin, je remercie tout les étudiants que j'ai eu la chance d'encadrer au cours de ces trois années, avec une mention spéciale pour Matthieu AUDEMARD et Elyes FEKI, dont la curiosité et le volontarisme constituent l'une des plus belle surprise de cette première expérience d'enseignement.

Sur un plan personnel enfin, je tiens à remercier non-exhaustivement, et sans notion d'ordre ou de hiérarchie (c'est à dire conformément à mes aspirations) : Lili, qui m'a porté et supporté tout au long de la rédaction de cette thèse. J'aime à croire que le pire est passé, et que le meilleur reste à vivre (et quand bien même, j'ai adoré le pire !); Mes parents, qui doivent contempler, un sourire ironique aux lèvres, les conséquences de ma lamentable tentative d'échapper à un avenir scientifique. Comme quelqu'un qui se reconnaîtra me l'a fait jadis remarquer, ils ont construit, par la gratuité de leur amour, une maison qui me protège des averses; Béa, qui devait être la seule personne plus émue que moi au moment de l'énoncé du verdict. Déjà 13 ans que nos histoires se côtoient, s'influencent, s'éloignent et se rapprochent, et je n'en vois ni n'en présage la fin; Nolwenn, qui a su remettre du désordre et de la légèreté dans une vie déjà bien entropique, me

semblait-il. Cette petite promenade au royaume des illusions romanesques ne fut-elle pas tout simplement exquise ?; Émilie, pour m'avoir enseigné un peu de sa belgitude, et que la mélancolie est un sentiment noble; Ma petite soeur préférée, que je ne désespère pas de voir grandir d'un peu plus près dans un avenir proche. Puisse-je t'offrir ma foi en ton bonheur futur; Mon p'tit frère préféré (trop facile, je sais), pour m'avoir offert pendant tant d'années l'occasion de méditer sur la politique, le cul et le foot, quotidiennement au cours de nos debriefings cathodiques à des horaires impossibles. Je lui souhaite une prompte intégration dans le monde de ceux qui écoutent France-Info avant d'aller au travail et qui sentent de la bouche (l'entreprise, quoi ...), quoique je doute sincèrement que sa fréquentation satisfasse à ses aspirations humanistes (et le syndicalisme, sinon ?); Toyo, Ludo, David et Nico, avec qui on aura érigé la défaite en art de vivre, quoi qu'il paraisse finalement bien impossible d'expliquer en quoi cet art procède d'un humanisme au pays du can-doisme (mais pourtant, c'en est bien un ...), ni pourquoi cette aspiration aura au final rendue possible la musique qu'on a créée ensemble. Non, on va VRAIMENT pas y arriver ! J'espère revenir vite et souvent pour voir vos petites gueules, et entendre vos nouveaux gimmicks; Enfin Tof, Mélanie, Alina, Alexander et Nazli, les copains de cordées avec qui on a déjà vécu des quantités d'expériences assez impressionnantes, propices à faire vieillir un peu vite, parfois, mais en bonne compagnie ! Merci d'avoir tous participé à la création d'un lieu itinérant (la pièce où nous sommes, le caillou auquel on est plus ou moins suspendus) où il fait bon manger, boire, vivre, et grimper, oui, surtout grimper !

2. INTRODUCTION

Il est loin désormais, ce temps où le biologiste moléculaire, ayant chaussé ses bésicles et ravivé le feu en prévision d'une nouvelle nuit sans sommeil, s'attelait à la recherche d'une incongruité dans ce listing du génome d'*Haemophilus Influenzae*, trônant sans partage sur le bureau bancal... Peut-être n'a-t-il d'ailleurs jamais existé ailleurs que dans l'imaginaire des jeunes bioinformaticiens, élevés avec Internet, et bercés d'illusion par les témoignages épiques de quelques *anciens*, pionniers de la grande époque ? Que celui qui ne fut pas interloqué à la vue d'un biologiste, cherchant un ARNt *de visu* dans une séquence d'ADN, me jette la première pierre... Quoi qu'il en soit, ce temps est désormais révolu, avec la croissance exponentielle des données disponibles, et nous vivons actuellement la fin d'une transition entre un traitement *manuel* des données issues de la biologie moléculaire et son automatisation par le biais de l'informatique. L'élaboration de tels dispositifs de traitement est assurément une mission centrale de la bioinformatique, champ émergeant dont on serait bien en peine de définir les limites. Comment concilier en effet, dans une définition sans exception, le travail d'organisation rationnelle et de mise à disposition des données (Expasy,Embl,UniProt), l'analyse automatisée de données de symptômes, destinée à identifier automatiquement les facteurs d'une maladie inconnue, ou encore la recherche de la conformation la plus stable pour une macromolécule dans un modèle thermodynamique ? Tout au plus pourra-t-on proposer l'utilisation d'un ordinateur, permettant l'automatisation du traitement de l'information, comme dénominateur commun à toutes ces méthodes.

L'automatisation des traitements nécessite une étape de formalisation préalable. Quelle est la manifestation dans nos données d'intérêt du phénomène recherché, et dans quel langage la décrire ? C'est entre autre cette nécessaire formalisation qui a provoqué l'apparition des modèles dans les méthodes bioinformatiques. Celles-ci ne focalisaient ainsi plus sur des comparaisons deux-à-deux de nos objets de prédilection (séquences, structures, données d'expression), mais cherchaient désormais à capturer les propriétés communes à un ensemble de ces objets par le biais d'un modèle. Dans un premier temps, des modèles se sont développés parallèlement dans une acception strictement probabiliste et dans un sens de la théorie des langages, comme les expressions régulières ou les grammaires non-contextuelles (stochastiques). En effet, celles-ci permettent la capture de phénomènes liés à des proximités séquentielle et/ou spatiale, propriété souhaitable dans l'optique d'une application à la génomique.

Quand l'apparition d'un phénomène connu est suspectée dans un objet d'intérêt, une question qu'il est naturel de se poser est : *L'apparition de ce phénomène est elle imputable au hasard ?* On peut utiliser pour illustrer ce principe le célèbre paradoxe du singe et de la machine à écrire :

Supposons qu'on attache un singe immortel et insensible à l'ennui à une machine à écrire, alors il finira par écrire *Hamlet* de Shakespeare avec probabilité 1 (ou n'importe quel autre livre). La raison de ce phénomène, rarement constaté en pratique car n'arrivant en moyenne que toutes les 26^{130000} frappes[1], est liée au fait que la frappe *au hasard* de touches sans mémoire attribue une probabilité non-nulle à toute séquence. Transposé dans le domaine du vivant, on peut, dans une approximation grossière, considérer le génome d'une espèce comme une *chaîne de Markov*, c'est à dire que la probabilité d'une base en une position donnée ne dépend que d'un sous ensemble des bases précédentes. Dans une version non dégénérée, une telle chaîne attribuera donc une probabilité non-nulle à la présence en son sein de chaque motif de taille inférieure à celle du génome[2].
Plus généralement, on utilise pour répondre à la question initiale le concept statistique de *significativité*, concept comparable à une quantification de l'étonnement face à un phénomène observé. Celui-ci est fréquemment matérialisé dans la bioinformatique par les valeurs du Z-score et de la P-value. Ils sont respectivement calculés à partir des formules

$$\text{Z-Score}(m) = \frac{m - \mu_0}{\sigma_0} \qquad \text{P-Value}(m) = \mathbb{P}(X_0 \geq m \mid H_0)$$

où m est la valeur observée pour un paramètre p, H_0 est le modèle *nul*, la variable aléatoire X correspondant à la valeur du paramètre p dans H_0. Enfin μ_0 et σ_0 sont respectivement la moyenne et l'écart-type associés à la variable X dans H_0.

Ces deux indicateurs de la significativité d'un phénomène observé (ou d'autres proches) sont omniprésents dans les démarches bioinformatiques. Ils permettent d'attirer l'attention des biologistes expérimentaux sur des phénomènes encore incompris ou en tout cas non-induits par les connaissances avérées. Une telle application fait alors apparaître la nécessité de modèles nuls complexes, aptes à tenir compte de l'existant. En effet, il est important, au moment d'évaluer la significativité d'un phénomène observé, de tenir compte de tous les phénomènes connus oeuvrant sur les objets considérés, et donc de leur impact sur ces objets. Par exemple, on peut tenir pour significative la rareté du motif TGA dans une séquence intergénique. Cette rareté sera, du moins partiellement, expliquée et donc non-signifiante au sein d'une région codante du fait de l'interdiction d'un tel triplet en phase codante trop proche d'un START. Avec la complexité croissante des phénomènes validés, et leur multiplication au sein de mêmes régions, il est essentiel de construire des modèles nuls *capturant* l'impact de *tous les phénomènes avérés*, et qui feront donc ressortir par complément ceux encore inexpliqués et méritant une étude. Il existe des classes de modèles pour lesquels l'analyse en vue de l'évaluation d'un test est facile ou, du moins, déjà résolue analytiquement ou algorithmiquement. On dispose alors de formules closes, comme pour la probabilité d'apparition d'une séquence sous un modèle de Markov, ou encore d'algorithmes pour évaluer les critères de significativité. Cependant dans de nombreux cas, les phénomènes dont on doit tester la présence dans le cadre du test sont trop complexes pour bénéficier d'une

[1] Dans le modèle uniforme et en négligeant l'auto-corrélation du texte.

[2] Évidemment, ce modèle est imparfait, car les mutations en certaines positions se retrouvent de fait interdites car létales et intransmissibles. Le modèle de Markov montre ici ses limite, car étant théoriquement interdit de prendre en compte d'avantage d'information, comme la position exacte, par exemple.

approche purement analytique. En particulier, dans le domaine des motifs structurés, notre capacité d'analyse symbolique se limite pour l'instant aux expressions régulières [66]. Au-delà des expressions régulières, on peut parfois trouver des décompositions qui rendent possible la génération aléatoire des objets modélisés. Une évaluation probabiliste par une méthode de type Monte-Carlo de la significativité d'une découverte est alors possible. Pour cela, on engendre des séquences ou des structures aléatoires, de façon à calculer la valeur moyenne d'un paramètre, par exemple. En comparant la valeur observée pour un paramètre avec celle ainsi obtenue en moyenne dans le modèle, on obtient une indication sur la significativité du phénomène observé.

On opère une distinction entre les modèles statistiques et modèles syntaxiques. Les modèles statistiques associent des probabilités, en général non-nulles, à chacun des objets sur un espace préalablement défini, comme l'ensemble des séquences sur {A,C,G,T} pour l'ADN. Les modèles syntaxiques discriminent dans l'espace de départ les objets autorisés (ou acceptés) de ceux interdits (ou rejetés). Parmi les premiers, on trouve les modèles markoviens, omniprésents en bioinformatique et qui permettent la prise en compte de dépendances séquentielles entre les bases. Les seconds comprennent, entre autre, les motifs PROSITE [8] ou toute autre expression régulière permettant d'exprimer un consensus séquentiel. Un mariage entre ces deux types de formalismes peut avoir lieu, par exemple afin d'exprimer la présence de motifs structurés dans des séquences présentant des contraintes statistiques (ARNt dans un organismes riches en GC, par exemple). On qualifiera les modèles cumulant l'interdiction de certains objets et l'attribution de probabilités aux autres de modèles statistico-syntaxiques. Plusieurs classes de modèles méritant ce qualificatif voient le jour, principalement dans l'optique d'application à la recherche de motifs. Parmi celles ci, les profils HMM et les grammaires stochastiques, qu'on présente brièvement en vue de leur génération aléatoire. On présentera aussi les SVG qui excèdent de loin les précédentes au niveau *expressivité*, mais ne présentent pas d'aspects statistiques à proprement parler:

- Profils HMM : Un profil HMM est un consensus, c'est à dire l'expression probabiliste condensée d'un ensemble de motifs, basée sur une chaîne de Markov cachée (Hidden Markov Model). Les profils HMM sont principalement utilisés pour modéliser des familles dans les bases de données dédiées aux protéines [91; 8]. Une introduction plus générale aux HMM peut être trouvée dans un des ouvrages de référence concernant l'analyse de séquences biologiques, du à Durbin *et al* [37]. Cependant, ces profils sont d'une expressivité, au sens de la théorie des langages, trop faible pour capturer certains types d'interactions. En effet, leur pouvoir expressif est exactement égal à celui des expressions régulières, auquel serait adjoint un modèle de Markov.

- Grammaires stochastiques : Il s'agit de grammaires auxquelles on adjoint des probabilités associées à chacune des réécritures pour un non-terminal. Prises en tant que modèle génératif, elles ne permettent pas le contrôle simultané de la distribution de probabilité sur les structures engendrées et de l'espérance de la taille de celles-ci. Or la maîtrise de la taille est essentielle à qui veut *mimer le réel*, surtout dans ces formalismes aptes à mod-

éliser des interactions longue-distance. En pratique, l' espérance de la taille des objets engendrée diverge, et une génération, par rejet anticipée, est au mieux exponentielle, à moins de pratiquer une génération dite *de Boltzmann*, qui contraint alors totalement les probabilités d'émission.

- String Variable Grammars : Les SVG [86] (Et les Grammaires à Transformation Morphiques de Sinoquet [89]) sont basées sur un principe d'unification de variables, dont on attend qu'elle apparaissent plus loin dans le texte, identiquement ou ayant subies un morphisme et/ou quelques altérations dans le cas des GTM. Elles sont quant à elles extrêmement expressives, dépassant même l'expressivité des grammaires non-contextuelles pour décrire une partie du *contextuel*. Elles peuvent ainsi modéliser des phénomènes complexes d'un point de vue algorithmique, comme les pseudo-noeuds (Parenthèsages $A.B.A'.B'$). Malheureusement, cette expressivité d'ordre supérieur induit des difficultés algorithmiques dans le cadre de la génération aléatoire. Principalement, la présence dans ces grammaires de variables non-instanciées et de taille variables introduit une ambiguïté dans la dérivation d'un mot. Cette ambiguïté, recherchée par le créateur de ce formalisme afin de trouver plusieurs instances d'un motif dans un texte, met en échec les approches classiques de génération aléatoire.

Il n'existait donc pas encore de modèle qui soit suffisamment expressif pour capturer des phénomènes non-strictement séquentiels, comme les interactions à longue-distance, et permette la génération aléatoire contrôlée.

C'est pourquoi nous avons étudié, dans la première partie de cette thèse, les propriétés d'une adjonction *statistique* aux modèles basés sur des grammaires non-contextuelles, et dû à Denise et al. [29], les grammaires pondérées. Celles-ci, de même que les grammaires classiques sur lesquelles elles sont basées, restreignent leur langage à un sous-ensemble des séquences sur le vocabulaire considéré. Cela permet, par le biais d'un codage ou d'une bijection, de décrire des objets structurés, comme les arbres, les structures secondaires d'ARN ou encore l'ensemble des séquences décrites par un consensus. On n'évoquera plus désormais que la génération de séquences, en gardant à l'esprit que les séquences en question peuvent être des codages pour des objets plus complexes. En outre, ces grammaires associent à chaque lettre un poids, contributif multiplicativement à la probabilité d'émission d'une séquence la contenant, ce qui permet d'altérer la distribution des séquences.

Un algorithme de génération aléatoire, en temps linéaire après un précalcul en temps et espace quadratique sur la taille de la séquence engendrée, existait déjà pour ce formalisme avec des poids donnés. On propose une adaptation de la méthode dite de Boltzmann, due à Duchon et al [36], aux grammaires pondérées, qui évite désormais ce précalcul et permet même la génération en temps linéaire quand la taille des séquences à engendrer est *relâchée* d'un facteur ε. De plus, nous prouvons la possibilité de réaliser avec une grammaire pondérée la conjugaison d'un modèle de Markov et d'une grammaire non-contextuelle. Pour cela, nous exhibons une construction pour une telle grammaire.

L'altération des probabilités d'émission liées à l'ajout de pondérations peut aussi être util-

isée pour contraindre les fréquences des différents symboles dans les séquences engendrées. Après avoir proposé des pistes et algorithmes pour l'évaluation les fréquences associées à une pondération donnée, nous dérivons un algorithme probabiliste, basé sur une technique d'optimisation due à Vanden Berghen [14] pour le problème inverse, à savoir la détermination d'une pondération réalisant des fréquences imposées.

Enfin, la découverte de contraintes multiples pesant sur des ensembles de séquences soulève la nécessité de prendre simultanément en compte des phénomènes capturés par des modèles différents, sous peine de s'*étonner* de l'apparition de phénomènes *émergeants*. On a donc besoin d'un langage de modèles expressifs pour distinguer l'émergeant du non-encore compris. C'est à ce besoin que souhaite répondre `GenRGenS`, une boîte à outils de génération aléatoire dans de différentes classes de modèles de séquences aléatoires, incluant le formalisme des grammaires pondérées. Nous illustrons l'utilité des différentes classes de modèles sur des exemples issus de problématiques biologiques.

Dans une deuxième partie, nous nous intéressons à la structure secondaire de l'ARN. Celle-ci, dans son acception classiquement admise par les bioinformaticiens, représente une sorte de frontière dans les problèmes liés à sa comparaison, à sa résolution *in silico*, à la recherche de motifs structurés... Elle fait l'objet d'un regain d'intérêt de la part de la communauté biologique. En effet, avec la croissante popularité de l'hypothèse de *monde à ARN*, les promesses liées à l'ARN-interférence ou la compréhension du rôle de la structure dans les mécanismes traductionnels non-conventionnels, la structure de l'ARN cesse d'être considérée uniquement dans l'ARN ribosomal et de transfert, et est désormais considérée comme informative aussi dans l'ARN messager. Dans un premier temps, nous proposons quelques rappels de biologie moléculaire, suivis d'une description des grandes classes d'ARN *structurés*.

Nous proposons alors une modélisation de la structure secondaire d'ARN par des grammaires non-contextuelles. Ces grammaires sont équivalentes du point de vue de l'ensemble de structures engendrées, mais peuvent s'avérer plus ou moins aptes à capturer différentes contraintes pouvant peser sur ces séquences. Nous ajoutons ensuite des pondérations, obtenant ainsi des grammaires pondérées permettant la contrainte des nombres et tailles de sous-structures. Nous obtenons, dans un modèle simple distinguant uniquement les bases appariées et non-appariées, une évaluation de la pondération en fonction des fréquences attendues. Nous obtenons aussi un résultat similaire, et remarquablement simple, pour la grammaire pondérée des tiges-boucles. Enfin, nous appliquons l'algorithme de recherche des pondérations à une grammaire de l'ARNt, ainsi qu'à la grammaire générale afin d'obtenir des pondérations dans le cas du 23s ribosomal d'eucaryote et du 5s ribosomal bactérien.

Enfin, le besoin de statistiques structurelles fiables, lié à la modélisation des ARN par des grammaires pondérées nous a amené à nous poser le problème de la *planarisation* de structure d'ARN. En effet, la définition pour la structure secondaire d'ARN qu'on a adopté lors de cette modélisation exclut les pseudo-noeuds, c'est à dire des arcs en situation de croisement dans une représentation circulaire de l'ARN. Or l'extraction d'un ensemble de liaisons à partir d'un ARN cristallographié, et qui peut être réalisée par les logiciels MC-Sym [55] ou RNAView [104], nous retourne des pseudo-noeuds. On remarque aussi la présence de telles sous-structures dans les

structures obtenues par une approche comparative [22]. C'est pourquoi on s'est intéressé au problème de la planarisation de structure d'ARN. On propose un algorithme de programmation dynamique en $\mathcal{O}(\min(m,n)m)$ pour la maximisation pondérée du nombre d'arcs. Un lien avec le problème Max-Independant-Set dans les graphes circulaires révèle une complexité concurrentielle avec les meilleurs algorithmes pour ce problème.

La troisième partie est consacrées à l'étude et l'utilisation d'un modèle de séquence afin d'étudier la sensibilité d'algorithmes de recherche de similarités heuristique. Les études consacrées à l'évaluation de la sensibilité des algorithmes heuristiques ont été menées conjointement avec L. Noé et G. Kucherov. Les aspects énumératifs des chemins culminants font actuellement l'objet d'une étude, en collaboration avec M. Bousquet-Mélou. Il s'agit des chemins *culminants*, séquences encodant l'évolution du score d'homologie de deux séquences au cours de sa construction par l'heuristique X-Drop de BLAST [4]. On peut voir ces chemins comme des marches aléatoires de taille bornée, composées de pas $(1,a)$ et $(1,-b)$ correspondant respectivement au bonus attribué sur une conservation (encodée par m) et au malus lié à une altération (encodée par $\overline{m}$). L'interrogation à l'origine de ces travaux portait sur la sensibilité associée à une graine, c'est à dire la probabilité d'observer une séquence donnée sur $\{m,\overline{m}\}^*$ dans un chemin culminant. L'optimalité du score d'alignement impose le respect, par la marche qui lui est associée, des deux propriétés suivantes : La *positivité*, car une marche atteignant des ordonnées négatives verra son score optimisé par son suffixe débutant à une des ordonnées négatives; La *culminance*, est la propriété d'atteindre en son dernier pas la plus grande hauteur jamais atteinte. J'ai étudié les propriétés des langages associés aux chemins culminants, et il s'est avéré que ceux ci n'appartiennent pas à la classe des langages non-contextuels. Il n'était donc pas possible d'utiliser une telle grammaire pour modéliser ces chemins.

A partir d'une relation de récurrence entre les suffixes associés à ces chemins, il a été possible de mettre en oeuvre l'approche récursive introduite par Wilf [103], qui a résulté en un algorithme en temps linéaire après un précalcul en $\mathcal{O}(n^3)$ opérations arithmétiques. La récurrence ainsi trouvée a alors permis à mes coauteurs, L. Noé et G. Kucherov, de découvrir une variante de l'algorithme de Keich, ce qui a permis l'évaluation de la sensibilité exacte associée à une graine [53]. On a ainsi mis en évidence un biais dans les modèles utilisés par les précédentes approches.

Mais certaines questions restaient encore sans réponse. En particulier, on avait tout d'abord tenté une génération par rejet à partir de $\{m,\overline{m}\}^n$, génération qui revient à tirer à pile ou face n fois et rejeter la séquence obtenue tant qu'elle ne code pas un chemin culminant valide. Cette génération aléatoire semblait de complexité linéaire quand $a > b$, quadratique quand $a = b$ et exponentielle quand $a < b$. Comment pouvions nous expliquer ce phénomène, et en tirer partie pour obtenir un algorithme de génération aléatoire efficace ? Ces aspects ont été étudiés en collaboration avec M. Bousquet-Mélou, et ont permis l'obtention d'algorithmes de rejets linéaires à partir de différents langages.

Partie I

ETUDE ET ENRICHISSEMENT DES MODÈLES PRIS EN CHARGE PAR GenRGenS

Plan et contributions

Le premier chapitre de cette thèse est consacré à un nouveau formalisme introduit par Denise et al. [29], les **grammaires pondérées**, ainsi qu'à leur génération aléatoire efficace au sein du logiciel `GenRGenS`. Après quelques rappels et définitions théoriques (Section 3.1, page 25), suivis d'un état de l'art de la génération aléatoire de séquences à partir d'une grammaire non-contextuelle (Section 3.3, page 33), nous abordons l'étude des grammaires pondérées. Nous rappelons l'adaptation de l'approche récursive proposée dans Denise et al[29] (Section 4.1, page 47), que nous complétons par une adaptation de la méthode dite de Boltzmann[36] au cas des grammaires pondérées (Section 4.2, page 48). Nous nous intéressons alors au gain d'expressivité lié à l'adjonction d'une pondération à une grammaire, et proposons une méthode pour simuler par une grammaire pondérée le produit d'un modèle de Markov et d'une grammaire non-contextuelle (Section 4.3, page 50).

Ensuite, nous nous intéressons à la relation entre les pondérations et les fréquences des différents symboles terminaux (les lettres des séquences modélisées). En particulier, nous souhaitons calculer les pondérations réalisant des fréquences attendues pour les terminaux, car par exemple observées dans des séquences biologiques. Les outils et méthodes permettant de résoudre théoriquement ce problème se heurtant malheureusement en pratique à des difficultés liées au calcul symbolique, nous nous intéressons à une résolution heuristique, par le biais d'un algorithme d'optimisation, de ce problème. Un tel algorithme nécessite la résolution du problème inverse, c'est à dire le calcul des fréquences moyennes des terminaux pour des pondérations données, c'est pourquoi nous proposons un algorithme basé sur une approche récursive pour ce problème (Section 4.4.2, page 55). La mise en conjonction de cet algorithme avec un algorithme d'optimisation dû à F. Vanden Berghen [14] donne naissance au prototype `GrgFreqs` qui permet l'obtention des pondérations associées à des fréquences attendues dans une grammaire non-contextuelle donnée (Section 4.4.3, page 59).

Enfin, nous présentons `GenRGenS`, un logiciel dédié à la génération aléatoire de séquences génomiques, et proposant des générateurs génériques pour différentes classes de modèles. Parmi ceux-ci, les modèles de Markov (homogènes, hétérogène *cadré* ou cachés *multi-urnes*), les expressions régulières, motifs PROSITE, grammaires non-contextuelles uniformes ou pondérées et les modèles *hiérarchiques* (Section 5, page 62).

Contributions :

- Adaptation de la méthode de Boltzmann aux grammaires pondérées
  Section 4.2, page 48
- Simulation par une grammaire pondérée d'un *produit* de modèle de Markov et de modèle à base de grammaire non-contextuelle
  Section 4.3, page 50
- Optimisation de la génération de séquences à partir d'une grammaire non-contextuelle ayant une composition *exacte* imposée
  Section 4.4.1, page 53
- Calcul exact des fréquences des symboles terminaux pour une taille et une pondération données
  Section 4.4.2, page 55
- Conception et implémentation du prototype `GrgFreqs` calculant les pondérations associées à des fréquences attendues
  Section 4.4.3, page 59
- Conception et diffusion de `GenRGenS`, un logiciel dédié à la génération aléatoire de séquences génomiques
  Section 5, page 62

Notations

Conventions typographiques	
$\mathscr{A}, \mathscr{B}, \mathscr{C} \ldots$	Les lettres manuscrites sont généralement réservées aux ensembles.
$S, T, U \ldots$	Symboles non-terminaux ou éléments d'une spécification combinatoire.
$\mathrm{s}_n, \mathrm{t}_{n'} \ldots$	Variables contenant les cardinaux des restrictions de l'ensemble des mots dérivables de $S, T, \ldots$ à ceux de taille $n, n', \ldots$ Par exemple, $\mathrm{s}_n \Leftrightarrow \mathscr{L}(S)_n$.
Mathématiques	
$\{a, b, c \ldots\}$	Un ensemble comprenant des éléments a, b, c, $\ldots$.
$\lvert\mathscr{A}\rvert$	Cardinal (nombre d'éléments) de l'ensemble $\mathscr{A}$.
$\mathscr{A}_n$	Restriction de l'ensemble $\mathscr{A}$ à ses objets de taille n.
$\sum_{x \in \mathscr{A}} f(x)$	Somme sur tous les éléments x de l'ensemble $\mathscr{A}$ de la valeur $f(x)$, pour f une fonction quelconque.
$\prod_{x \in \mathscr{A}} f(x)$	Produit sur tous les éléments x de l'ensemble $\mathscr{A}$ de la valeur $f(x)$, pour f une fonction quelconque. Le produit peut être commutatif (multiplication) ou non-commutatif (concaténation).
$\sum_{x \in \mathscr{A}} f(x)$	Union de tous les ensembles $f(x)$, pour tout x élément de l'ensemble $\mathscr{A}$, et f une fonction à valeur dans l'ensemble des ensembles.
$\mathscr{O}(f(n))$	Classe des fonctions croissant **au plus aussi vite** qu'une fonction f donnée. Formellement, pour toute fonction g dans $\mathscr{O}(f(n))$, $\lim_{n\to\infty} g(n)/f(n) = \kappa$, où $\kappa > 0$ est une constante de n.
$\Theta(f(n))$	Classe des fonctions croissant **aussi vite** qu'une fonction f donnée. Formellement, pour toute fonction g dans $\mathscr{O}(f(n))$, $\lim_{n\to\infty} g(n)/f(n) = \kappa$, où $\kappa \geq 0$ est une constante de n.
Théorie des langages	
ω	Séquence ou mot.
ω_i	Symbole apparaissant en i-ème position dans ω.
$\lvert\omega\rvert$	Taille ou longueur de la séquence ω.
$\lvert\omega\rvert_\alpha$	Nombre d'occurrences du symbole α dans ω.
$\overline{\omega}$	*Image-miroir* du mot ω.
$t, t', t_1 \ldots$	Symboles terminaux.
Σ	Vocabulaire terminal, c'est à dire l'alphabet qu'utilisent les séquences.
$S, S', S_1 \ldots$	Symboles non-terminaux.
$\mathscr{N}$	Vocabulaire non-terminal, ensemble des non-terminaux.
$\mathscr{L}(S)$	Langage des mots obtenus à partir de S.
$\mathscr{L}^n$	Langage des mots obtenus par concaténation de n éléments de $\mathscr{L}$.
$\mathscr{L}^*$	Langage des mots obtenus par concaténation d'un nombre arbitraire (éventuellement nul) d'éléments de $\mathscr{L}$.
Combinatoire énumérative	
n	Taille des séquences d'intérêt.
$S(z)$	Série génératrice du langage associé au non-terminal S. Formellement,. $S(z) = \sum_{\omega \in \mathscr{L}(S)} z^{\lvert\omega\rvert} = \sum_{i \geq 0} \mathrm{s}_i z^i$.
$[z^n] f(z)$	Coefficient de degrés n dans le développement de f en $z = 0$. .

3. TECHNIQUES DE GÉNÉRATION ALÉATOIRE UNIFORME : LE CAS NON-CONTEXTUEL

3.1 Outils et lexique

Nous présentons ici quelques outils et techniques utilisés dans la suite de ce document. La plupart sont rattachés à la théorie des langages ou à la combinatoire, énumérative ou analytique.

3.1.1 Grammaires formelles

Les grammaires non-contextuelles apparaissent lors de l'établissement de la hiérarchie des langage formels de Chomsky-Schützenberger. Un langage formel est un ensemble potentiellement infini de séquences (ou *mots*) sur un alphabet. Une grammaire est un mécanisme de réécriture à base de règles qui engendrent, par dérivations successives à partir d'un symbole initial, un langage. La position d'un langage dans la hiérarchie de Chomsky-Schützenberger dépend du type des règles nécessaires pour engendrer exactement ce langage :

- Type 3 : Les grammaires régulières :
  Leurs règles sont toutes de forme $S \to tN$ ou $S \to t$ (ou toutes de forme $S \to Nt$ ou $S \to t$, mais pas de cohabitation possible de ces deux types de règle au sein d'une même grammaire régulière). Ces règles suffisent pour engendrer les langages reconnus par un *automate fini*, ou ceux dénotés par une *expression régulière*.

- Type 2 : Les grammaires *hors-contexte* ou *algébriques* :
  Leurs règles sont de la forme $S \to \alpha$ où α est une séquence non contrainte de symboles terminaux ou non-terminaux. Ces grammaires permettent de décrire les langages *hors-contexte*, qui sont reconnus par des *automates à pile*.

- Type 1 : Les grammaires *contextuelles* :
  Elles admettent des règles de la forme $\alpha S\beta \to \alpha\eta\beta$ où α, β et η sont des séquences non-contraintes de symboles. De telles grammaires permettent d'engendrer les langages *contextuels*, reconnus par des machines de Türing travaillant sur mémoire linéaire sur la taille du mot en entrée.

- Type 0 : Les grammaires *générales* :
  Toutes les règles $\alpha \to \beta$, où α et β sont des séquences de symboles, sont autorisées. Toute la puissance expressive de la *machine de Türing* est nécessaire pour reconnaître les langages construits à partir de cette grammaire, une telle machine ne s'arrêtant pas nécessairement sur de telles entrées.

Les règles étant de façon évidente d'expressivités strictement croissantes, il est donc possible de mimer une grammaire utilisant des règles d'une catégorie donnée avec une grammaire utilisant des règles d'une catégorie supérieure. Cette hiérarchie des grammaires induit donc une hiérarchie sur les langages formels ainsi engendrés. On retrouve en outre cette hiérarchie dans les modèles de calculs nécessaires pour tester l'appartenance d'un mot à un langage. Cette hiérarchie se retrouve aussi dans la complexité des algorithmes nécessaires pour engendrer uniformément un mot du langage.

3.1.2 Modèles syntaxiques/modèles statistiques

Les langages formels induisent ce qu'on est tenté d'appeler des *modèles syntaxiques* sur les séquences génomiques. Parmi l'ensemble des séquences sur un alphabet donné (A,C,G,U pour l'ARN), ils ne font que distinguer les séquences *autorisées* des *interdites*. En l'absence d'informations supplémentaires, on est forcé de supposer que les séquences *autorisées* sont équiprobables, soit parmi les séquences d'une taille donnée, soit parmi l'ensemble des séquences *autorisées*. C'est pourquoi de nombreux formalismes se sont attachés à adjoindre une distribution de probabilité aux séquences *autorisées*.

Les chaînes de Markov peuvent par exemple être utilisées dans une version dégénérée, en annulant certaines probabilités de transition, ce qui interdit alors l'apparition de certains motifs de tailles supérieures à l'ordre. Cependant, l'expressivité réduite sur le plan syntaxique de ce formalisme[1] permet uniquement l'adjonction de contraintes simples, du type *motif exclu*, au modèle. Celles-ci peuvent en outre avoir un impact non-négligeable sur la distribution des séquences.

Les grammaires stochastiques de Searls semblent conjuguer l'expressivité des grammaires non-contextuelles et la puissance d'une distribution de probabilité non-uniforme. De plus, les paramètres du modèles peuvent être appris à partir d'un ensemble de séquences par une stratégie *maximiser la vraisemblance*. Malheureusement, ce formalisme n'offre que peu de prise sur l'espérance de la taille des séquences engendrées, et est donc peu adapté à la génération aléatoire.

Nous nous intéressons donc à un autre formalisme, introduit par Denise et al., les *grammaires non-contextuelles pondérées* qui, via l'introduction de la notion de pondérations, permet d'altérer la distribution des séquences et ce faisant de contraindre les espérances des différentes lettres dans les séquences générées.

3.2 Définitions formelles

3.2.1 Langages

Soit Σ^* le monoïde libre, c'est à dire l'ensemble constitué des k-uplets d'élément de Σ, $k \geq 0$, doté d'un produit . non-commutatif, associatif, tel que $a, b \in \Sigma^* \Rightarrow a.b \in \Sigma^*$, et enrichi d'un élément ε neutre pour le produit.
On appellera Σ le *vocabulaire* ou le *lexique*.
Les éléments de Σ sont alors appelés les *symboles*.
Les éléments de Σ^* sont quant à eux les *mots* ou *séquences*.
On note ε le *mot vide*, de taille nul et élément neutre pour le produit de concaténation.

Définition 1 (Langage) :
Un langage est un ensemble $\mathscr{L} \subseteq \Sigma^*$. □

[1] Le graphe des k-grams dans une chaîne de Markov étant aisément assimilable à un automate fini.

Définition 2 (Fonction de taille) :
On appelle fonction de taille la fonction $|.| : \Sigma^* \to \mathbb{N}$ *définie récursivement par :*

$$|\varepsilon| = 0 \tag{3.1}$$
$$|\alpha| = 1,\ \alpha \in \Sigma \tag{3.2}$$
$$|\omega.\omega'| = |\omega| + |\omega'|,\ \omega, \omega' \in \Sigma^* \tag{3.3}$$
$$\tag{3.4}$$

□

Si l'on ne considère que le nombre d'occurrences d'un symbole $\alpha \in \Sigma$, on obtient une fonction $|.|_\alpha : \Sigma^* \to \mathbb{N}$ telle que :

$$|\varepsilon|_\alpha = 0 \tag{3.5}$$
$$|\alpha|_\alpha = 1 \tag{3.6}$$
$$|\beta|_\alpha = 0,\ \beta \neq \alpha,\ \beta, \alpha \in \Sigma \tag{3.7}$$
$$|\omega.\omega'|_\alpha = |\omega|_\alpha + |\omega'|_\alpha,\ \omega, \omega' \in \Sigma^* \tag{3.8}$$
$$\tag{3.9}$$

3.2.2 Grammaires non-contextuelles

Définition 3 (Grammaire non contextuelle) :
Une grammaire non-contextuelle G est caractérisée par la donnée d'un quadruplet $(\Sigma, \mathcal{N}, X, \delta)$ *où :*

- Σ *est l'ensemble des symboles terminaux, parfois appelé* vocabulaire *ou* lexique.
- $\mathcal{N}$ *est l'ensemble des symboles non-terminaux.*
- $X \in \mathcal{N}$ *est le symbole non-terminal initial à toute dérivation, aussi appelé* axiome.
- $\delta \subseteq \mathcal{N} \times (\mathcal{N} \cup \Sigma)^*$ *est l'ensemble des règles de réécriture des non-terminaux de la grammaire, aussi appelées* productions.

□

Définition 4 (Dérivation directe) :
Soit $G = (\Sigma, \mathcal{N}, X, \delta)$*, on dit qu'un mot* $\beta \in (\mathcal{N} \cup \Sigma)^*$ dérive directement *dans G d'un mot* $\alpha \in (\mathcal{N} \cup \Sigma)^*$ *ssi :*

$$\alpha = \alpha_p S \alpha_s, \quad \beta = \alpha_p \omega \alpha_s \quad et \quad (S, \omega) \in \delta$$

On note cette propriété $\alpha \to_G \beta$.

□

Définition 5 (Dérivation) :
Dans $G = (\Sigma, \mathcal{N}, X, \delta)$*, un mot* $\omega \in \Sigma^*$ dérive *d'un non-terminal* $S \in \mathcal{N}$*, ssi il existe* $(s_1, \ldots, s_k) \in (\mathcal{N} \cup \Sigma)^*, k \in \mathbb{N}$ *tels que :*

$$S \to_G s_1 \to_G \ldots \to_G s_k \to_G \omega$$

Définition 6 (Langage associé à une grammaire) :
On appelle langage d'un non terminal S *dans* G, *qu'on note* $\mathscr{L}_G(S)$, *l'ensemble des mots terminaux dérivables à partir du non-terminal* S *dans* G :

$$\mathscr{L}_G(S) = \{\omega \in \Sigma^* | S \rightarrow_G \omega\}$$

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

$$\mathscr{L}(G) = \mathscr{L}_G(X)$$

Définition 7 (Grammaire non-ambiguë) :
Une grammaire non-contextuelle G *est dite* non-ambiguë *si tout mot* ω *de* $\mathscr{L}(G)$ *admet une unique dérivation à partir de l'axiome* X, *au cours de laquelle le non-terminal le plus à gauche est réécrit à chaque étape.* □

Dans la suite du document et dans un souci de simplification des notations, on omettra d'indicer les dérivations et langages par la grammaire considérée quand celle ci est implicite.
D'autre part, l'expression $\alpha \rightarrow \beta$ est équivalente au couple $(\alpha, \beta) \in \delta$.
De plus, on notera $\mathscr{R}(\delta, S)$ l'ensemble des règles de δ ayant le non-terminal S comme partie gauche :

$$\mathscr{R}(\delta, S) = \{(S' \rightarrow \alpha) \subseteq \delta | S' = S\}$$

Exemple :
Considérons la grammaire P des *palindromes* composés de lettres a ou b. Les palindromes sont des mots dont la lecture est identique *de gauche à droite* ou *de droite à gauche*.

- $\Sigma = \{a, b\}$, il s'agit des lettres présentes dans les mots à engendrer.
- $\mathscr{N} = \{X\}$ est l'ensemble des symboles non-terminaux, réduit au seul axiome dans ce cas précis.
- X est l'axiome.
- L'ensemble δ des règles est le suivant :

$$\begin{array}{c|lcl} \delta_1 & X & \rightarrow & aXa \\ \delta_2 & X & \rightarrow & bXb \\ \delta_3 & X & \rightarrow & \varepsilon \end{array}$$

Le mot $aabbaabbaa$ appartient au langage $\mathscr{L}(P)$ décrit par la grammaire, car on peut le dériver de l'axiome X de la façon suivante :

$$X \overset{\delta_1}{\rightarrow} aXa \overset{\delta_1}{\rightarrow} aaXaa \overset{\delta_2}{\rightarrow} aabXbaa \overset{\delta_2}{\rightarrow} aabbXbbaa \overset{\delta_1}{\rightarrow} aabbaXabbaa \overset{\delta_3}{\rightarrow} aabbaabbaa$$

Dans [29], Denise et al. introduisent le formalisme des grammaires non contextuelles pondérées au niveau des terminaux (le terme *grammaire pondérée* apparaissant déjà dans la communauté *traitement du langage*), dans lequel la grammaire se voit adjoindre une fonction de pondération.

Définition 8 (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 :

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

Définition 9 (Grammaire non-contextuelle pondérée) :
Une grammaire non-contextuelle pondérée *est caractérisée par un quintuplet* $(\Sigma, \mathscr{N}, X, \delta, \pi)$ *où :*

- Σ *est l'ensemble des symboles terminaux.*
- $\mathscr{N}$ *est l'ensemble des symboles non-terminaux.*
- $X \in \mathscr{N}$ *est l'axiome.*
- $\delta \subseteq \mathscr{N} \times (\mathscr{N} \cup \Sigma)^*$ *est l'ensemble des règles.*
- $\pi : \Sigma \to \mathbb{R}$ *est une fonction de pondération.* □

L'intérêt d'une telle adjonction n'apparaît pleinement que dans l'optique d'une génération aléatoire non-uniforme et sera détaillé en section 3.3.

3.2.3 Classe combinatoire

Définition 10 (Classe combinatoire) :
Une classe combinatoire $\mathscr{C}$ *est un ensemble muni d'une fonction de* taille $|.| : \mathscr{C} \to \mathbb{N}$, *qui induit des sous ensembles* **finis** $\mathscr{C}_n$ *d'objets de* $\mathscr{C}$ *ayant taille* n. □

Si une classe combinatoire est engendrée par un ensemble Ψ de fonctions d'arités variées, alors on parle de classe combinatoire *décomposable*. On appelle alors Ψ l'ensemble des *constructeurs* de $\mathscr{C}$.
De plus, on notera $|\psi|$, $\psi \in \Psi$ l'arité d'un constructeur, c'est à dire son nombre d'arguments. Cependant, ces constructeurs ne travaillent pas uniquement sur des objets de $\mathscr{C}$, mais sur des objets antécédents par l'application de $f \in \Psi$ d'objets de $\mathscr{C}$.

Définition 11 (Antécédents d'une classe combinatoire) :
L'ensemble $\mathscr{C}^\triangleleft$ *des antécédents d'une classe combinatoire* $\mathscr{C}$ *relativement à un ensemble* Ψ *de constructeurs est défini récursivement par :*

$$\mathscr{C}^\triangleleft = \{\alpha = \psi(\beta),\ \psi \in \Psi \text{ et } \beta \in \mathscr{C}^{\triangleleft|\psi|}\}$$

Exemple :
Les langages $\mathscr{L}$ forment des classes combinatoires assez naturelles, car la fonction de taille $|.|$: $\Sigma^* \to \mathbb{N}$ définie usuellementhabituellement par $|\varepsilon| = 0$, $|t| = 1$, $\forall t \in \Sigma$ et $|\alpha.\beta| = |\alpha| + |\beta|$ sur le monoïde libre Σ^* est bien telle que $\mathscr{L}_n \leq |\Sigma|^n$ soit fini.
En particulier, on peut s'intéresser pour illustrer les définitions ci-dessus au langage $\mathscr{P}$ des mots $a^n.b^n$, décrit par la grammaire suivante :

$$P \to aPb \mid \varepsilon$$

Soit ϕ_1 et ϕ_2 les morphismes définis par $\phi_1(a) = \phi_2(a) = a$, $\phi_1(b) = \phi_2(b) = b$, $\phi_1(P) = aPb$ et $\phi_2(P) = \varepsilon$. Les constructeurs ϕ_1 et ϕ_2 naturellement associés à cette classe sont les extensions multiplicatives des morphismes ψ_1 et ψ_2 définis par

$$\psi_1(\alpha.\beta) = \phi_1(\alpha).\psi_1(\beta),\ \alpha \in \{a,b,P\} \quad \psi_2(\alpha.\beta) = \phi_2(\alpha).\psi_2(\beta),\ \alpha \in \{a,b,P\}$$

On a alors $\mathscr{P}_{2k} = \{a^k.b^k\}$ et $\mathscr{P}^{\lhd} = \mathscr{P} \cup \{a^k P b^k,\ k \in \mathbb{N}\}$.
Les antécédents d'une classe combinatoire peut être vus comme des objets *immatures* faisant encore apparaître des *traits de construction* dans les cas où la construction des objets n'est pas *récursive pure* comme les spécifications de [42]. Cette définition nous permettra de définir formellement les univers sur lesquels agiront les *générateurs*.

3.2.4 Analyse combinatoire

Commençons par définir l'objet de base de la combinatoire analytique, la série génératrice.

Définition 12 (Série génératrice de dénombrement) :
Soit $\mathscr{C}$ une classe combinatoire et $c_n = |\mathscr{C}_n|$ le nombre d'objets de taille n dans $\mathscr{C}$, alors on appelle série génératrice de dénombrement de $\mathscr{C}$ la série formelle $C(z)$ en une variable complexe $z \in \mathbb{C}$ telle que

$$C(z) = \Sigma_{o \in \mathscr{C}} z^{|o|} = \Sigma_{n \geq 0} c_n z^n$$ □

Ces séries sont centrales à bien des démarches d'analyse combinatoire, car elles peuvent être construites quasi-automatiquement à partir d'une *spécification* non ambiguë pour la classe combinatoire considérée. En particulier, dans le cas d'une grammaire non-ambiguë, l'ensemble des règles de la grammaire peut être vu comme une telle spécification, et se transpose alors directement en un système d'équations sur les séries génératrices associées aux non-terminaux. Il s'agit de la méthodologie DSV, baptisée ainsi par X.G. Viennot conformément au souhait de M. Schützenberger.

Définition 13 (Méthodologie DSV) :
Soit $\mathscr{C}$ une classe combinatoire, la méthodologie DSV est une méthode générale pour la détermination de la série génératrice associée à la classe $\mathscr{C}$. Elle consiste en l'application des 3 points ci-dessous :

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

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

$$\begin{cases} S_1 & \to & \alpha_1^1 \mid \ldots \mid \alpha_1^{m_1} \\ S_2 & \to & \alpha_2^1 \mid \ldots \mid \alpha_2^{m_2} \\ & \cdots & \\ S_k & \to & \alpha_k^1 \mid \ldots \mid \alpha_k^{m_k} \end{cases} \Rightarrow \begin{cases} S_1(z) & = & \phi(\alpha_1^1) + \ldots + \phi(\alpha_1^{m_1}) \\ S_2(z) & = & \phi(\alpha_2^1) + \ldots + \phi(\alpha_2^{m_1}) \\ & \cdots & \\ S_k(z) & = & \phi(\alpha_k^1) + \ldots + \phi(\alpha_k^{m_k}) \end{cases}$$

  où $k = |\Sigma|$, $S_i \in \mathscr{N}$, $\alpha_j^i \in (\mathscr{N} \cup \Sigma)^*$, $S_j(z) = \sum_{o \in \mathscr{L}(S_j)} z^{|o|}$ et ϕ est telle que

$$\begin{aligned} \phi(\varepsilon) &= \varepsilon \\ \phi(t\,\omega) &= z.\phi(\omega),\ t \in \Sigma \\ \phi(S_i\,\omega) &= S_i(z).\phi(\omega),\ S_i \in \mathscr{N} \end{aligned}$$

□

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

$$S_\alpha(z) = \mathscr{L}(X)(z) = \mathscr{L}(\mathscr{G})(z) = \mathscr{L}(z) = \mathscr{C}(z)$$

Définition 14 (Coefficient d'une série génératrice) :
Soit $C(z)$ la série génératrice de dénombrement d'une classe $\mathscr{C}$. On appelle coefficient de degré *n de $C(z)$, qu'on note $[z^n]C(z)$, le coefficient du monôme z^n dans un développement de $C(z)$ autour de $z = 0$. En remarquant qu'un tel développement est aussi un développement de Taylor, on a*

$$[z^n]C(z) = \frac{1}{n!}\frac{\partial^n C(0)}{\partial z} = \mathrm{c}_n$$

où c_n est le nombre d'objets de taille n dans $\mathscr{C}$. □

On voit alors une première application possible des séries génératrices à l'étude combinatoire d'une classe $\mathscr{C}$. En effet, si la série génératrice se prête aux dérivations multiples, alors on peut en déduire des expressions exactes pour les c_n. On peut dériver ainsi les nombres de Catalan de la série génératrice des mots de Dyck.

Il existe en outre des outils théoriques, comme l'*inversion de Langrange*, pour extraire les coefficients de séries dont la série génératrice n'est pas fournie explicitement, mais naît d'une composition de fonction.

Une deuxième application des séries génératrice concerne l'asymptotique des coefficients. Pour de nombreuses familles de classes combinatoires (rationnelles, algébriques ou holonome), il existe des comportements asymptotiques *stéréotypés* pour les coefficients. Le dogme central de l'analyse de singularités est en effet le suivant :

- *La singularité de plus petit module ρ, ou* singularité dominante*, d'une série génératrice détermine le comportement exponentiel des coefficients.*
- *Le type de singularité détermine le comportement sous-exponentiel (polynomial, poly-logarithmique, ...).*
- *Les constantes multiplicatives du comportement asymptotique sont déduites du comportement de la série au voisinage de ρ.*

On n'entrera pas dans les détails, rappelant juste le rapport marquant entre singularité dominante et ordre exponentiel.

Définition 15 (Ordre exponentiel) :
Soit $\{c_n\}$ une suite à coefficients dans $\mathbb{R}$.
On appelle ordre exponentiel de $\{c_n\}$ la fonction K^n telle que :

$$c_n \asymp K^n$$

C'est à dire $|c_n| = \vartheta(n)K^n$ où $\vartheta(n)$ est un facteur subexponentiel :

$$\lim_{n\to\infty} \sup |\vartheta(n)|^{\frac{1}{n}} = 1$$

Théorème 1 (Ordre exponentielle des coefficients d'une série) :
Soit $C(z) = \sum_{n\geq 0} c_n z^n$ une fonction analytique en 0 et ρ le module de la singularité dominante de C.
Alors :

$$c_n \asymp \left(\frac{1}{\rho}\right)^n$$

Par exemple, dans le cas d'un langage algébrique, cela signifie qu'on pourra dériver directement le comportement exponentiel des nombres de mots de taille n de la plus petite valeur ρ pour z qui *pose problème*.

3.3 Principes de génération aléatoire

La génération aléatoire uniforme d'objets combinatoires a donné lieu a de nombreuses études. En particulier dans le cas des grammaires non-contextuelles, deux applications de la méthode récursive donnent des algorithmes de génération aléatoire en taille exacte [42; 45; 30] effectuant un nombre d'opérations arithmétiques linéaires $\mathcal{O}(n\log(n))$ (resp. $\mathcal{O}(n\log(n))$ et $\mathcal{O}(n^{1+\varepsilon})$) après un précalcul en $\mathcal{O}(n^2)$ (resp. $\mathcal{O}(n)$) en utilisant une mémoire $\mathcal{O}(n^2)$ (resp. $\mathcal{O}(1)$ et $\mathcal{O}(n^{1+\varepsilon})$). La méthode dite de Boltzmann permet la génération en taille approchée en temps $\mathcal{O}(n)$ sans autre précalcul que celui, symbolique, d'une série génératrice. Des explications plus détaillées sur ces algorithmes peuvent être trouvées dans la partie 3.3.3. Nous nous intéressons ici à la transposition de ces méthodes au cas des grammaires non-contextuelles pondérées pour une génération aléatoire non uniforme maîtrisée.

3.3.1 Complexité arithmétique ou complexité binaire ?

Il est d'usage d'évaluer la complexité d'un algorithme en le nombre d'opérations atomiques nécessaires à son exécution. Traditionnellement, on choisit des opérations arithmétiques comme unité de l'évaluation de la complexité. On parle alors de *complexité arithmétique*, l'hypothèse implicite étant que de telles opérations s'effectuent en un temps constant, ou du moins borné par une constante.

Malheureusement, quand les nombres manipulés explosent, cette hypothèse devient irréaliste. En particulier, si les nombres faisant l'objet de ces opérations sont en $\mathcal{O}(k^n)$ sur un paramètre n, le simple codage d'un de ces nombres nécessitera un espace proportionnel à n.

De telles explosions sont fréquentes dans les algorithmes de génération aléatoire, il est alors plus pertinent de s'intéresser à la *complexité binaire*. La *complexité binaire* est le nombre d'opérations nécessaires à l'algorithme sur un processeur travaillant sur des nombres au codage borné (32-bits).

On indiquera donc par une astérisque (*) les complexités arithmétiques nécessitant la prise en compte de la taille du codage des nombres manipulés lors de l'évaluation de la complexité binaire. La différence entre ces deux complexités porte alors sur les facteurs suivants en fonction du type d'opération pour des nombres nécessitant un codage de taille n :

- Comparaison : $\Theta(n)$ au pire et $\mathcal{O}(1)$ en moyenne.
  On compare deux entiers p et q codés sur n bits en regardant les termes du poids fort au poids faible. Dès que les termes p_i et q_i diffèrent, on peut conclure. Or, dans l'hypothèse d'une distribution uniforme des nombres comparés, $\mathbb{P}(p_i = q_i) = \frac{1}{2}$, $\forall i \in [1, n]$ et soit Y le nombre de comparaisons binaires :

$$\mathbb{P}(Y = k) = (1 - \mathbb{P}(p_{k+1} = q_{k+1})) \prod_{i=1}^{k} \mathbb{P}(p_i = q_i) = \frac{1}{2^{k+1}}$$

L'espérance $\mathbb{E}(Y)$ du nombre de comparaisons binaires est donnée par :

$$\begin{aligned}\mathbb{E}(Y) &= \sum_{i=0}^{n}(i+1)\mathbb{P}(Y=k)\\ &= \sum_{i=0}^{n}\frac{i+1}{2^{i+1}}\\ &= 2-\frac{n-1}{2^{n+1}}<2\end{aligned}$$

- Addition et soustraction : L'addition de l'écolier avec propagation de retenue a une complexité binaire en $\Theta(n)$ opérations binaires.
- Multiplication : L'utilisation de la transformée rapide de Fourier [85] permet d'obtenir le produit de deux nombres en $\mathcal{O}(n\log(n)\log\log(n))$ opérations binaires.

En général, il est raisonnable d'imaginer un *rapport linéaire* entre les *complexités binaire* et *arithmétique*, et on notera $\mathcal{O}(n^{k+\varepsilon})$ les complexités pour lesquelles des facteurs poly-logarithmiques ont été négligés.

3.3.2 Méthode par rejet

Soit $\mathscr{A}$ un ensemble dans lequel on veut tirer un élément aléatoirement des mots de taille n, mais pour lequel il est difficile de trouver une approche constructive (Ex. : Langage contextuels). Parfois, il est possible d'engendrer efficacement des objets dans un surensemble[2] $\mathscr{B}\supset\mathscr{A}$. On engendre donc des éléments de $\mathscr{B}_n$ jusqu'à obtenir un élément de $\mathscr{A}_n$. Les tirages correspondants aux différents éléments engendrés dans $\mathscr{B}_n$ étant indépendants, la probabilité d'obtenir un élément ω de $\mathscr{A}_n$ est donc égale à celle d'obtenir ω dans $\mathscr{B}_n$, renormalisée à 1. En particulier, dans le cas uniforme, ce facteur de renormalisation est égal à $|\mathscr{B}_n|/|\mathscr{A}_n|$ et la probabilité $\mathbb{P}(\omega \mid \mathscr{A}_n)$ d'émission d'un élément ω donné est alors égale à

$$\mathbb{P}(\omega \mid \mathscr{A}_n) = \mathbb{P}(\omega \mid \mathscr{B}_n)\frac{|\mathscr{B}_n|}{|\mathscr{A}_n|} = \frac{1}{|\mathscr{B}_n|}\frac{|\mathscr{B}_n|}{|\mathscr{A}_n|} = \frac{1}{|\mathscr{A}_n|}$$

La **complexité de cette méthode** repose sur plusieurs facteurs :

- Complexité $\alpha(n)$ de la génération dans $\mathscr{B}_n$.
- Coût $\beta(n)$ du test d'appartenance à $\mathscr{A}_n$.
- L'espérance $\mathbb{E}(X \mid \mathscr{A}_n, \mathscr{B}_n)$ du nombre X de rejets, tel que

$$\mathbb{E}(X \mid \mathscr{A}_n, \mathscr{B}_n) = \frac{1}{\sum_{\omega\in\mathscr{A}_n}\mathbb{P}(\omega \mid \mathscr{B}_n)}$$

[2] Ex. : Σ^* est par exemple un surensemble de tous les langages dans lequel la génération aléatoire de mots de taille n est aussi simple que n lancer de dés !

Dans le cas uniforme, cette expression se simplifie et on obtient

$$\mathbb{E}(X \mid \mathscr{A}_n, \mathscr{B}_n) = \frac{|\mathscr{B}_n|}{|\mathscr{A}_n|}$$

Au final, cette complexité revient à $(\alpha(n) + \beta(n))\mathbb{E}(X \mid \mathscr{A}_n, \mathscr{B}_n)$.

Dans un grand nombre de cas, le test d'appartenance à $\mathscr{A}_n$ peut être effectué au cours de la création de l'objet candidat, et donc $\beta(n)$ peut être négligé devant $\alpha(n)$. De plus, il n'est pas systématiquement nécessaire de disposer de l'intégralité de l'objet candidat pour décider de son appartenance à $\mathscr{A}_n$. On peut alors pratiquer un rejet *anticipé*, ou florentin, baptisé ainsi d'après l'école florentine de combinatoire à l'origine de cette technique. On peut ainsi, pour des objets combinatoires simples (chemins positifs, ...), obtenir des termes $(\alpha(n)+\beta(n))$ sub-linéaires, ce qui permet de compenser un terme $\mathbb{E}(X \mid \mathscr{A}_n, \mathscr{B}_n)$ non-constant. En particulier, Barcucci et al [11] arrivent ainsi à engendrer des préfixes de mots de Dyck en temps linéaire.

Cependant, cette technique n'est pas très efficace dans le cas des grammaires non-contextuels car, à moins de choisir judicieusement le surensemble, car le nombre de rejet est souvent exponentielle, et presque sûrement en $\mathscr{O}(n\sqrt{n})$ (D'après l'asymptotique *stéréotypée* en $\rho^{-n}/n\sqrt{n}$ de [34]) à partir de Σ^*.

3.3.3 Méthode récursive

Comme son nom l'indique, la méthode récursive repose sur l'existence d'une récurrence sur les nombres d'objets accessibles à partir d'un choix local. A partir de celle-ci, et au prix d'un précalcul, il est possible de déterminer avec quelles probabilités opérer ce choix local afin d'engendrer les objets d'une classe selon la distribution souhaitée. Un tel algorithme se décompose donc en deux phases :

- **Précalcul** : On calcule des valeurs (cardinaux, poids, fonction de partition) définies par récurrence pour chacun des choix locaux.

- **Génération** : On opère une série de poids locaux avec des probabilités judicieusement déduites des valeurs précalculées, et qui restreignent de plus en plus l'ensemble des futurs possibles (objets accessibles), jusqu'à obtenir le singleton ne contenant que l'objet engendré selon la distribution de probabilité souhaitée.

On trouve naturellement une telle récurrence dans la structure récursive qu'est une grammaire non-contextuelle. Cependant, confronté à une règle ayant pour partie droite de nombreux non-terminaux, on se pose la question de la répartition des lettres à engendrer parmi les différents non-terminaux. Une telle répartition, implémentée de façon naïve, risquerait d'être exponentielle sur le nombre maximal de non-terminaux apparaissant en partie droite d'une règle. C'est pourquoi on utilise une forme normale, la forme normale de Chomsky, qui fait apparaître au plus deux termes dans chaque partie droite de règle.

La forme normale de Chomsky : Décomposition canonique *des grammaires non-contextuelles*

Dans le cas général, la méthode récursive présuppose de l'existence d'une décomposition non-ambiguë de la classe combinatoire considérée. Dans le cas des grammaires non-contextuelles, une telle décomposition peut être trouvée dans la forme normale de Chomsky.

Définition 16 (Forme Normale de Chomsky (FNC)) :
Une grammaire $G = (\Sigma, \mathcal{N}, X, \delta)$ *est en* forme normale de Chomsky *ssi, pour tout* $S \in \mathcal{N}$ *l'ensemble* $\mathcal{R}(\sigma, S) = \{(S, \alpha) \in \delta\}$ *des règles ayant S pour partie gauche est tel que :*

$$\begin{aligned} & \mathcal{R}(\delta, S) = \{S \to t\},\ t \in \Sigma \\ ou\quad & \mathcal{R}(\delta, S) = \{S \to S_1 S_2\},\ S_1, S_2 \in \mathcal{N} \\ ou\quad & \mathcal{R}(\delta, S) = \{(S \to S_1),\ (S \to S_2)\},\ S_1, S_2 \in \mathcal{N} \end{aligned}$$

Plus simplement, on pourra dire et noter que chaque non-terminal S d'une grammaire en forme normale de Chomsky n'est *concerné* que par une des trois situations ci-dessous :

$$\begin{array}{ll} \text{Règle Terminale :} & S \to t, \quad t \in \Sigma \\ \text{Règle Produit :} & S \to S_1 S_2, \quad S_1, S_2 \in \mathcal{N} \\ \text{Règle Union :} & S \to S_1 \mid S_2, \quad S_1, S_2 \in \mathcal{N} \end{array}$$

On notera $\mathcal{F}$ l'ensemble des grammaires en forme normale de Chomsky.

Proposition 1 (Existence de la forme normale de Chomsky) :
Pour toute grammaire G non-ambiguë, il existe une grammaire G' *en forme normale de Chomsky reconnaissant le même language, moins éventuellement le mot vide :*

$$\forall\, G, \exists\, G' \in \mathcal{F} \text{ telle que } \mathcal{L}(G') = \mathcal{L}(G) \setminus \{\varepsilon\}$$

On donne ci-dessous une idée de l'algorithme utilisé pour une telle mise en forme :

1. Binarisation des conjonctions
2. Restriction des productions ε à l'axiome
3. Suppression des règles unaires solitaires
4. Réécriture des règles impliquant des terminaux
5. Binarisation des disjonctions

On pourra trouver une preuve complète de l'existence d'une FNC dans [49].

L'ordre des opérations lors de la mise en forme normale de Chomsky devra toutefois respecter quelques contraintes, sous peine d'obtenir une grammaire en FNC de taille exponentielle par rapport à celle de la grammaire initiale.

Exemple :

Considérons les arbres ternaires, c'est à dire des arbres n'ayant que des noeuds de degré trois ou zéro. On encode naturellement les arbres ternaires de taille n par des mots de taille $3n$ issus de la grammaire suivante :

$$\begin{aligned} S &\rightarrow aSbScS \\ &\rightarrow \varepsilon \end{aligned}$$

Une approche *naïve* du dénombrement de tels mots conduit à la récurrence suivante sur les nombres s_n de mots issus de S de taille n :

$$\begin{aligned} \mathrm{s}_0 &= 1 \\ \mathrm{s}_1 &= 0 \\ \mathrm{s}_2 &= 0 \\ \mathrm{s}_n &= \sum_{i=0}^{n-3} \sum_{j=0}^{n-3-i} \mathrm{s}_i \mathrm{s}_j \mathrm{s}_{n-3-i-j},\ \forall n > 2 \end{aligned}$$

Le terme général de cette récurrence semble indiquer **une complexité quadratique** sur n pour l'évaluation du terme s_n à partir des termes de rangs inférieurs, c'est à dire **une complexité cubique** pour l'évaluation *ab initio* du terme s_n. Cependant, une mise en forme normale de Chomsky aboutit à la grammaire suivante :

$$\begin{aligned} S' &\rightarrow T \mid \varepsilon \\ T &\rightarrow MU \\ U &\rightarrow NO \\ M &\rightarrow V \mid A \\ N &\rightarrow W \mid B \\ O &\rightarrow X \mid C \\ V &\rightarrow AT \\ W &\rightarrow BT \\ X &\rightarrow CT \\ A &\rightarrow a \\ B &\rightarrow b \\ C &\rightarrow c \end{aligned}$$

Cette grammaire engendre bien le langage des arbres ternaires à partir de l'axiome S', en effet :

$$\begin{aligned} & T \rightarrow MU \\ \Leftrightarrow\ & T \rightarrow MNO \\ \Leftrightarrow\ & T \rightarrow (V \mid A)(W \mid B)(X \mid C) \\ \Leftrightarrow\ & T \rightarrow (AT \mid A)(BT \mid B)(CT \mid C) \\ \Leftrightarrow\ & T \rightarrow a(T \mid \varepsilon)b(T \mid \varepsilon)c(T \mid \varepsilon) \\ \Leftrightarrow\ & T \rightarrow aS'bS'cS' \\ \Leftrightarrow\ & S' \rightarrow aS'bS'cS' \mid \varepsilon \end{aligned}$$

On retrouve ici, à un renommage de S' en S près, la grammaire des arbres ternaires. Cette représentation donne naissance à une nouvelle récurrence sur les coefficient s_n qui, bien qu'apparemment plus complexe que la précédente du fait de la croissance du nombre de non-terminaux, ne fait apparaître que des produits de convolutions à deux termes. Cela permet un **calcul en temps linéaire** d'un coefficient s_n à partir de tous les coefficients de rang inférieur, ce qui ramène à une **complexité quadratique** en opération arithmétique le calcul *ab initio* de s_n.

$$
\begin{aligned}
\alpha_0 &= \begin{cases} 1 & \text{Si } \alpha = \mathrm{s} \\ 0 & \text{Sinon} \end{cases} \\
\alpha_1 &= \begin{cases} 1 & \text{Si } \alpha \in \{\mathrm{a},\mathrm{b},\mathrm{c},\mathrm{m},\mathrm{n},\mathrm{o}\} \\ 0 & \text{Sinon} \end{cases} \\
\alpha_2 &= 0,\ \forall \alpha \in \mathcal{N} \\
\mathrm{t}_n &= \textstyle\sum_{i=1}^{n-1} \mathrm{m}_i \mathrm{u}_{n-i} \\
\mathrm{u}_n &= \textstyle\sum_{i=1}^{n-1} \mathrm{s}_i \mathrm{o}_{n-i} \\
\mathrm{m}_n &= \mathrm{s}_n = \mathrm{o}_n = \mathrm{v}_n = \mathrm{w}_n = \mathrm{x}_n = \mathrm{t}_{n-1} \\
\mathrm{s}'_n &= \mathrm{t}_n
\end{aligned}
$$

Une telle forme normale sera un allié de poids quand nous serons confrontés au problème de la génération aléatoire de mots issus d'une grammaire, c'est à dire bien souvent au comptage de tels mots. En effet, une telle construction, préservant la non-ambiguïté supposée de la grammaire originale, se traduit immédiatement en des mécanismes de construction du langage pris en tant que classe combinatoire. De tels mécanismes peuvent bien sûr être trouvés dans la grammaire originale, mais induisent une complexité supérieure dans la spécification combinatoire, et amplifient les difficultés rencontrées lors de l'énumération, comme l'illustre l'exemple précédent.

Génération aléatoire de mots d'un langage non-contextuel[29; 42]

À partir d'une forme normale de Chomsky, il est classique de dégager des relations entre les cardinalités des langages associés aux non-terminaux (voir par exemple [29]).

Théorème 2 :
Dans une grammaire $\mathcal{G}$ non-ambiguë en FNC , les langages $\mathcal{L}(S)$ et les nombres s_k de mots de

taille k issus de S obéissent aux relations suivantes :

$$
\begin{array}{lll}
\textit{Règles Unions :} & S \to S' \mid S'' & \\
& \Rightarrow \quad \mathscr{L}(S) = \mathscr{L}(S') \cup \mathscr{L}(S'') & \\
& \Rightarrow \quad \mathrm{s}_k = \mathrm{s}'_k + \mathrm{s}''_k &
\end{array}
$$

$$
\begin{array}{lll}
\textit{Règles Produits :} & S \to S'S'' & \\
& \Rightarrow \quad \mathscr{L}(S) = \mathscr{L}(S').\mathscr{L}(S'') & \\
& \Rightarrow \quad \mathrm{s}_k = \sum_{i=1}^{k-1} \mathrm{s}'_i \mathrm{s}''_{k-i} &
\end{array}
$$

$$
\begin{array}{lll}
\textit{Règles Terminales :} & S \to t & \\
& \Rightarrow \quad \mathscr{L}(S) = \{t\} & \\
& \Rightarrow \quad \mathrm{s}_k = \begin{cases} 1 & \textit{Si } k = 1 \\ 0 & \textit{Sinon} \end{cases} &
\end{array}
$$

Dans le cas des règles produits, on notera que la somme exclut les termes $i = 0$ et $i = n$, de façon à éviter l'apparition d'un terme s'_n dans la somme. Un tel terme peut être éliminé sans remords car il est non-contributif à la somme. En effet, tout terme s'_n contribue au sein d'une multiplication avec s''_0, et $\mathrm{s}_0 = 0$, comme conséquence de la mise en forme normale de Chomsky.

Les relations portant sur les langages correspondent à la définition 6 du langage associé à un non-terminal. Afin de les transposer en relations sur les cardinalités, il faut, dans le cas des règles unions, que les langages associés aux deux alternatives soient disjoints. Il faut en outre que tout mot ω engendré par un non-terminal produit n'admette qu'une seule décomposition $\omega = u.v$, $u \in \mathscr{L}(S')$ et $v \in \mathscr{L}(S'')$. Ces deux conditions sont garanties car nécessaires à la non-ambiguïté de la grammaire.

Une fois le précalcul des coefficients s_k effectués, la génération aléatoire uniforme est immédiate :

Théorème 3 :
L'algorithme 1 engendre des objets de $\mathscr{L}(\mathscr{G})$ de taille n avec probabilité uniforme $\frac{1}{|\mathscr{L}(\mathscr{G})_n|}$.

PREUVE : Conformément à la philosophie générale de la méthode récursive formalisée par Wilf [103], l'algorithme opère une série de choix locaux entre des ensembles distincts d'objets. On peut donc raisonner par induction sur le nombre d'appels à la fonction generation de l'algorithme.On remarque que celui ci est bornée par $n|\mathscr{U}|$, où $\mathscr{U}$ est l'ensemble des non-terminaux de type *union*. En effet, il ne peut pas exister de cycle dans le graphe de dépendance des *unions* (non-ambiguïté). De plus, une génération ne faisant appel qu'à des non-terminaux de type *produit* voit son nombre d'appels borné par $n-1$, car l'arbre des appels est alors un arbre binaire à n feuilles (règles terminales).

D'abord, on remarque la correction de l'algorithme dans le cas des règles terminales, pour lesquelles le mot constitué d'un seul terminal renvoyé est bien le seul du langage. On suppose

Algorithme 1 Approche récursive pour la génération uniforme de mots d'un langage non-contextuel

Function generation($S \in \mathcal{N}$ **,** $k \in \mathbb{S}$**) :**
si $\mathscr{R}(S) = \{S \to t\}$ **alors**
  return t
sinon si $\mathscr{R}(S) = \{S \to S' \mid S''\}$ **alors**
  Tirer aléatoirement $x \in \mathbb{R}$ compris entre 0 et 1 exclu
  si $x < \frac{\mathrm{s}'_k}{\mathrm{s}_k}$ **alors**
    return generation(S', k)
  sinon
    return generation(S'', k)
  fin si
sinon si $\mathscr{R}(S) = \{S \to S'S''\}$ **alors**
  Choisir $k' \in [1, k-1]$ avec probabilité $\frac{\mathrm{s}'_{k'}\mathrm{s}''_{k-k'}}{\mathrm{s}_k}$
  return generation(S', k').generation$(S'', k-k')$
fin si

ensuite l'algorithme correct pour tout nombre d'appel $\alpha \leq p$, c'est à dire :

$$\forall S \in \mathcal{N},\ \forall n \in \mathbb{S},\ \forall \omega \in \mathscr{L}(S)_n,\ \mathbb{P}(\omega) = \frac{1}{|\mathscr{L}(S)_n|} = \frac{1}{\mathrm{s}_n}$$

Pour $\alpha = p+1$:

- $\mathscr{R}(S) = \{S \to S' \mid S''\}$
  Si S'(resp. S'') est choisi, alors la probabilité *a posteriori* d'un mot ω issu de S'(resp. S'') est $\mathbb{P}(\omega) = \frac{\mathrm{s}'_n}{\mathrm{s}_n}\frac{1}{\mathrm{s}'_n} = \frac{1}{\mathrm{s}_n}$ (resp. $\mathbb{P}(\omega) = \frac{\mathrm{s}''_n}{\mathrm{s}_n}\frac{1}{\mathrm{s}''_n} = \frac{1}{\mathrm{s}_n}$) par application de l'hypothèse de récurrence à l'appel sur S'(resp. S'').

- $\mathscr{R}(S) = \{S \to S'S''\}$
  Soit n' le nombre choisi, et $\omega = \omega'.\omega''$ le mot finalement engendré, où ω' (resp. ω'') est le mot de taille n'(resp. $n-n'$) engendré à partir de S'(resp. S'') avec probabilité $\frac{1}{\mathrm{s}'_{n'}}$(resp. $\frac{1}{\mathrm{s}''_{n-n'}}$), par application de l'hypothèse de récurrence. La non-ambiguïté de la grammaire implique qu'il n'existe qu'une décomposition de ω en produit de mots issus de S' et S''. On a donc *a posteriori* :
  $$\mathbb{P}(\omega) = \frac{\mathrm{s}'_{n'}\mathrm{s}''_{n-n'}}{\mathrm{s}_n}\frac{1}{\mathrm{s}'_{n'}}\frac{1}{\mathrm{s}''_{n-n'}} = \frac{1}{\mathrm{s}_n}$$

L'algorithme 1 engendre bien uniformément des objets de $\mathscr{L}(\mathscr{G})$ de taille n. ■

Complexité de la méthode

Nous décrirons d'abord la complexité de la méthode récursive dans une implémentation naïve, nous évoquerons ensuite diverses améliorations portant sur les différentes phases de l'algorithme.

Implémentation *naïve* :
La phase de précalcul, correspondant à l'implémentation des équations du théorème 2, est en $\mathcal{O}(n^2)^*$ opérations arithmétiques, nécessaires lors du calcul des produits. Comme évoqué en section 3.3.1, il faut multiplier cette complexité **arithmétique** par le coût de chacune de ces opérations, effectuée en précision arbitraire, ce qui entraîne une complexité **binaire** en $\mathcal{O}(n^{3+\varepsilon})$.

La complexité mémoire du précalcul est en $\mathcal{O}(n|\mathcal{N}|)^*$ variables encodant des nombres croissant généralement en $\mathcal{O}(|\Sigma|^n)$, et nécessitant donc chacune $\mathcal{O}(n)$ bits. La complexité mémoire de cette méthode est donc en $\mathcal{O}(n^2|\mathcal{N}|)$ bits.

La phase de génération nécessite la répartition des symboles à engendrer entre les deux termes d'une règle de type *produit*, ce qui peut requérir $\mathcal{O}(n)$ comparaisons, précédées pour chacune d'une multiplication de grands nombres en $\mathcal{O}(n\log(n)\log\log(n))$. De plus, on a vu que le nombre d'appels à la fonction `engendrer` de l'algorithme 1 est en $\mathcal{O}(|\mathcal{N}|n)$. La complexité en temps de la phase de génération est donc en $\mathcal{O}(|\mathcal{N}|n^{3+\varepsilon})$.

Optimisations :

- **Stratégie *boustrophédon*[42]:**
  Une organisation judicieuse des tests des candidats pour k' pour les non-terminaux *produits* permet de réduire à $\mathcal{O}(n\log(n))^*$ la complexité de la phase de génération. Précisément, on teste les valeurs putatives pour k' dans l'ordre $1, k-1, 2, k-2, \ldots, i, k-i, \ldots$ Le coût $f(n)$ au pire de la génération d'un mot de taille n à partir d'une arborescence de règles *produits* obéit donc à une récurrence de type

  $$f(n) = \max_{k' \in [1,n-1]} (f(k') + f(n-k') + 2\min(k', n-k'))$$

  dont la solution est en $\mathcal{O}(n\log(n))^*$.

- **Récurrences linéaires des coefficients de séries holonomes [45] :**
  Un résultat fort sur les séries holonomes [92], auxquelles appartiennent les séries génératrices des langages algébriques, conclut sur l'existence d'une récurrence linéaire à coefficients polynômiaux pour s_k les nombres de mots de taille k issus de S, $\forall S \in \mathcal{N}$, et mettant en jeu un nombre borné de termes :

  $$\mathrm{s}_k = P_1(n)\mathrm{s}_{k-1} + P_2(n)\mathrm{s}_{k-2} + \ldots + P_h(n)\mathrm{s}_{k-h}$$

  Il est donc possible d'évaluer les s_n, $\forall S \in \mathcal{N}$ en temps $\mathcal{O}(n|\mathcal{N}|)^*$ et en mémoire $\mathcal{O}(|\mathcal{N}|)^*$ en *oubliant définitivement* les termes antérieurs à $k-h$ au moment de calculer le k-ième terme.
  Au cours de la phase de génération, on peut *redescendre* en inversant la récurrence

  $$\mathrm{s}_{k-h} = \frac{\mathrm{s}_k - \left(P_1(n)\mathrm{s}_{k-1} + \ldots + P_{h-1}(n)\mathrm{s}_{k-(h-1)}\right)}{P_h(n)}.$$

  On traite le cas des règles *produit* en utilisant à la fois la récurrence inversée et les termes initiaux de la récurrence, pour chacun des deux non-terminaux mis en jeu. La mise en

conjonction de cette remarque avec une stratégie *boustrophédon* permet alors de limiter le surplus de calcul à cette étape. On obtient donc alors un algorithme de génération en $\mathcal{O}(n\log(n)|\mathcal{N}|)^*$.
Il est à noter que la récurrence permettant ce précalcul peut être extraite du système d'équation défini par les règles de la grammaire grâce à un outil de calcul symbolique comme `Gfun` [83].

- **Précalcul des produits [61] :**
  Partant du principe que ce sont les résolutions des produits qui empêchent l'obtention d'algorithme linéaires sur la génération, Mairson précalcule dans [61] les séries de produits de convolution et les stocke dans une structure de donnée adaptée. Il en résulte un surcoût lors de la phase de dénombrement, qui passe en $\mathcal{O}(n^2)^*$, et qui permet une génération en $\mathcal{O}(n)^*$.

- **Arithmétique flottante et évaluation *paresseuse* [30] :**
  Après avoir évalué dans le cas des langages rationnels [26] la perte d'information liée à l'utilisation de nombre flottants, Denise et al. généralisent dans [30] l'utilisation de cette arithmétique et déterminent des *intervalles de confiance*, dans lesquels l'algorithme se comporte comme si les calculs sans arrondis avaient été effectués. Si les nombres aléatoires tirés sont hors de ces intervalles, alors on a recours au calcul exact(évaluation paresseuse). Après avoir exhibé une relation entre la taille de la mantisse et la taille de ces intervalles, ils obtiennent un algorithme de génération en $\mathcal{O}(n^{1+\varepsilon})$(Choix judicieux de la relation mantisse/taille de séquence) en moyenne après un précalcul en $\mathcal{O}(n^{1+\varepsilon})$ (Amélioration due à l'évaluation rapide [97] des coefficients d'une série génératrice algébrique [27]).

3.3.4 Méthode de Boltzmann pour la génération d'objets combinatoires [36]

La méthode dite de Boltzmann [36] apporte actuellement un nouveau souffle au domaine de la génération aléatoire d'objets décomposables, inspirant de nombreux travaux[43; 68; 12; 39] de part sa beauté conceptuelle et la simplicité de sa mise en oeuvre. Elle consiste à *relâcher* initialement la taille des objets à engendrer tout en maintenant l'équiprobabilité des objets au sein d'une classe de tailles. Plus précisément, elle affecte à chaque objet $\omega \in \mathscr{C}$ non-étiqueté une probabilité

$$\mathbb{P}_x(\omega) = \frac{x^{|\omega|}}{C(x)} \tag{3.10}$$

où x est un paramètre analytique affectant la performance du générateur, et $C(z) = \sum_{n\geq 0} c_n z^n$ la série génératrice de dénombrement de la classe combinatoire $\mathscr{C}$[3].

On engendre alors des objets jusqu'à ce qu'un objet d'un taille acceptable soit obtenu. Par acceptable on entend : Soit d'une taille fixée n, auquel cas on parle d'un générateur de Boltzmann en taille exacte; Soit une tolérance ε est accordée, et des objets de taille $[n(1-\varepsilon), n(1+\varepsilon)]$ sont acceptés, auquel cas on parle d'un générateur en taille approchée.

[3] On retrouve, en substituant $e^{-\beta}$ à x dans l'équation (3.10), la loi de Boltzmann bien connue en physique statistique, et à l'origine du nom de cette méthode.

Dans le cas des grammaires non-contextuelles, Duchon et al. consacrent un paragraphe à l'analyse des points suivants :

1. L'algorithme de génération aléatoire selon la probabilité de l'équation (3.10).
2. La stratégie de rejet utilisée.
3. Le choix d'une valeur pour le paramètre x.

Langages non-contextuels et implémentation du modèle de Boltzmann

Les langages non-contextuels peuvent être construits à partir d'ensembles singletons $t_1, t_2, \ldots, t_k$, d'un produit non-commutatif $\times$(concaténation) et d'une union disjointe $+$. Ces constructions se *plongent* très bien à la fois dans l'algèbre des séries génératrices et dans l'univers de la génération de Boltzmann. Ces *briques de bases* sont bien sûr équivalentes aux trois types de règles dans les grammaires en forme normale de Chosmky. On aura donc autant de générateurs de Boltzmann que de non-terminaux dans la grammaire initiale.

Algorithme 2 Générateur de Boltzmann pour le non-terminal Sd'une spécification algébrique

Function $\Gamma\mathscr{S}(x)$ **:**
si $S \to t$ **alors**
  return t
sinon si $S \to S' \mid S''$ **alors**
  Tirer aléatoirement $y \in \mathbb{R}$ compris entre 0 et 1 exclu
  si $x < \frac{S'(x)}{S(x)}$ **alors**
    return $\Gamma\mathscr{S}'(x)$
  sinon
    return $\Gamma\mathscr{S}''(x)$
  fin si
sinon si $S \to S'S''$ **alors**
  return $\Gamma\mathscr{S}'(x) \times \Gamma\mathscr{S}''(x)$
fin si

Théorème 4 :
L'algorithme 2 engendre chaque objet $\omega \in \mathscr{L}(S)$ *avec probabilité*

$$\mathbb{P}_x(\omega) = \frac{x^{|\omega|}}{S(x)}$$

On peut constater qu'une telle distribution affecte des probabilités égales à deux objets de même taille, elle est donc uniforme pour une classe de taille donnée.
Remarque : On a fait référence dans l'algorithme aux séries génératrices de langages $S(z)$, $S'(z)$ et $S''(z)$. Les valeurs de ces séries en x sont obtenues grâce à l'intervention divine d'un *oracle*, et supposée résolues par ailleurs, dans l'article fondateur [36]. Leur obtention représente le seul obstacle à l'automatisation totale de la méthode, bien que le package `Maple GFun` [83] sache se montrer particulièrement perspicace dans la résolution des systèmes d'équations algébriques.

Valeur du paramètre x et performances

On dispose donc d'un générateur qui engendre des mots d'un langage de différentes tailles, qu'on va rejeter jusqu'à obtenir un mot de taille acceptable. Mais comment limiter l'impact du rejet sur la complexité de cet algorithme ? C'est là qu'intervient le paramètre x, qui permettra, selon l'approche adoptée, de *décaler la moyenne* de la distribution des tailles engendrées ou même de *passer à la limite* afin d'augmenter les performances d'une stratégie de rejet anticipé. C'est cette deuxième méthode qui est privilégiée dans [36], mais nous commencerons par rappeler la première, qui illustre à notre sens d'avantage la philosophie sous-jacente à cette technique :

- Générateur classique de Boltzmann :
  Soit $\mathbb{E}_x(N)$ l'espérance de la taille N d'un objet de $\mathscr{L}(S)$ engendré par un générateur de Boltzmann, on a
  $$\mathbb{E}_x(N) = \sum_{\omega\in\mathscr{L}(S)} |\omega|\mathbb{P}_x(\omega) = \sum_{n\geq 0} n\mathrm{s}_n \frac{x^n}{S(x)} = x\frac{S'(x)}{S(x)}$$
  De même, on accède aux moments d'ordre supérieur en dérivant la série génératrice de probabilité :
  $$\frac{x^2S''(x)+xS'(x)}{S(x)} = \frac{\sum_{n\geq 0}\mathrm{s}_n n(n-1)x^{n-2}x^2 + \mathrm{s}_n n x^{n-1}x}{S(x)} = \sum_{n\geq 0}\mathrm{s}_n n^2\frac{x^n}{S(x)} = \mathbb{E}_x(N^2)$$
  Ainsi, on connaît désormais une relation entre x et la moyenne des tailles générées. On peut aussi vérifier que la variance n'est pas trop importante, c'est à dire qu'on sait quantifier la probabilité de tirer un mot d'une taille admissible. Ce n'est malheureusement pas toujours le cas pour les grammaires non-contextuelles, et leur génération efficace grâce à un générateur de Boltzmann nécessite souvent une modification de la spécification[4] afin de modifier le type de singularité de la série génératrice. Cette transformation s'automatise assez bien, à condition toutefois de limiter les tailles des parties droites de règles (Forme Normale de Chomsky), et rend possible une génération en taille approchée en $\mathscr{O}(n)$ et exacte en $\mathscr{O}(n^2)$ à partir d'une valeur de x adaptée à la taille souhaitée.

- Générateur polaire de Boltzmann :
  Comme on l'a vu dans le cas précédent, il existe une relation entre la valeur du paramètre x et l'espérance $\mathbb{E}_x(N)$ de la taille d'un objet engendré par le générateur. Quand x tend vers une valeur ρ appelée *singularité dominante* de la série génératrice, alors $\mathbb{E}_x(N)\to\infty$. En s'intéressant à la distribution de probabilité, on constate que $\mathbb{P}_\rho(N=n_0)$ décroît avec N. Si l'on *tronque* la distribution, on gratifie $N=n_0$ d'une probabilité non-exponentiellement négligeable.

[4] Il s'agit de l'opérateur de pointage, qui *marque* un des symboles terminaux dans chaque mot engendré. Il suffit alors d'*oublier* le caractère particulier de cette lettre dans le mot engendré pour obtenir un générateur uniforme. La transformation équivalente au pointage dans les séries génératrices étant la dérivation, on obtient une distribution des taille plus *pointée*, qui entraîne une génération plus efficace [36].

Au niveau algorithmique, cela revient simplement à stopper et relancer l'algorithme de réécriture quand l'objet en cours de génération dépasse la taille n_0 souhaitée, ce qui peut être réalisé grâce à une variable globale. Cette stratégie, aussi appelée *rejet anticipé* a été introduite pour une génération par rejet efficace dans le cas des animaux dirigés [11]. Il est à noter qu'un tel *élagage* provoque l'apparition d'un facteur de normalisation dans la formule donnant les probabilité d'émission.

On doit alors évaluer la probabilité associée à une taille n_0 dans la distribution *tronquée* en n_0, afin de déterminer la complexité de l'algorithme. Dans le cas des grammaires non-contextuelles, deux propriétés assez naturelles contraignent fortement l'asymptotique des coefficients, et nous permettent de répondre à cette question.

Définition 17 (Irréductibilité d'une grammaire) :
Le graphe de dépendance $\mathscr{G}$ d'une grammaire $G = (\Sigma, \mathscr{N}, X, \delta)$ est le graphe dont les sommets sont les non-terminaux $\mathscr{N}$ de G et dans lequel il existe un lien $(\alpha, \beta) \in \mathscr{N}^2$ ssi $(\alpha \rightarrow \ldots\beta\ldots) \in \delta$.
On dit qu'une grammaire $G = (\Sigma, \mathscr{N}, X, \delta)$ est irréductible *si son graphe de dépendance est* fortement connecté*, c'est à dire que tout sommet est accessible à partir de tout autre sommet.* □

Définition 18 (Périodicité d'une grammaire) :
Une grammaire $G = (\Sigma, \mathscr{N}, X, \delta)$ est dite périodique *si l'ensemble $\mathscr{I}$ des rangs i pour lesquels $s_i = 0$ est inclus dans une progression arithmétique.* □

Théorème 5 (Forme stéréotypée des s.g. de classes récursives[36]) :
Soit $G = (\Sigma, \mathscr{N}, X, \delta)$ une grammaire irréductible *et* apériodique*, alors la série génératrice $G(z) = \sum_{n\geq 0} s_n z^n$ est de la forme*

$$G(z) = G(\rho) - c_0(1 - z/\rho)^{\frac{1}{2}} + \mathscr{O}(1 - z/\rho) \tag{3.11}$$

où ρ est la singularité dominante de G, et $G(\rho) > 0$, $c_0 > 0$. L'asymptotique des s_n est alors donnée par

$$s_n = [z^n]G(z) = \frac{c_0\rho^{-n}}{2\sqrt{\pi}n^{3/2}}(1 + \mathscr{O}(\frac{1}{n})) \tag{3.12}$$

□

L'équation (3.12) permet d'établir des complexités moyennes pour les générateurs par rejet anticipés en $\mathscr{O}(n)$ pour une taille approchée, et en $\mathscr{O}(n^2)$ pour une taille exacte.

On préférera l'approche polaire à celle classique, car elle ne nécessite que l'évaluation des différentes séries génératrices, supposées résolues par ailleurs, en sa singularité dominante ρ. On trouvera dans [36] une discussion sur l'impact d'une l'évaluation en précision bornée sur l'uniformité. Une telle restriction rend possible l'implémentation d'un générateur de Boltzmann hors du cadre d'un logiciel de calcul symbolique, une fois cependant les séries génératrices calculées. De plus, elle ne nécessite pas d'examen critique du type de singularité.

4. APPLICATION AUX GRAMMAIRES NON-CONTEXTUELLES PONDÉRÉES

4.1 Génération récursive pondérée[29]

Dans [29], Denise et al. s'intéressent à une génération aléatoire non-uniforme de mots d'un langage non-contextuel $\mathscr{L}$. Leur but est de contraindre par ce biais les nombres d'occurrences de chacun de symboles terminaux, afin de simuler plus crédiblement certains phénomènes biologiques. Pour cela, ils fixent *a priori* les probabilités des différents mots $\omega \in \mathscr{L}$:

$$\mathbb{P}(\omega|\pi) = \frac{\pi(\omega)}{\sum_{\omega' \in \mathscr{L}} \pi(\omega')} \tag{4.1}$$

où π est la fonction de pondération, définie page 29 et étendue multiplicativement des symboles non-terminaux vers les mots de $\mathscr{L}$.

Théorème 6 :
Soit A_π l'adaptation de l'algorithme 1 page 40 obtenue en remplaçant

$$(S \to t) \Rightarrow \mathrm{s}_n = \begin{cases} 1 & \textit{Si } n = 1 \\ 0 & \textit{Sinon} \end{cases}$$

par

$$(S \to t) \Rightarrow \mathrm{s}_n = \begin{cases} \pi(t) & \textit{Si } n = 1 \\ 0 & \textit{Sinon} \end{cases}$$

lors de la phase de dénombrement préliminaire, les s_n devenant alors les sommes des poids des séquences de taille n issues des S, c'est à dire : $\mathrm{s}_n = \sum_{\omega \in \mathscr{L}(S)_n} \pi(\omega)$
Alors A_π engendre bien les mots de $\mathscr{L}(\mathscr{G})_n$ avec les probabilités décrites par l'équation (4.1).

PREUVE : Par récurrence sur k, on prouve que $\mathrm{s}_k = \sum_{\omega \in \mathscr{L}(S)_k} \pi(\omega)$. En effet, on vérifie trivialement cette propriété pour les règles de type $(S \to t)$. De plus, les appels récursifs reviennent à :

- $(S \to S'S'')$:

$$\begin{aligned} \mathrm{s}_k &= \sum_{i \in [1,k-1]} \mathrm{s}'_i \mathrm{s}''_{k-i} = \sum_{i \in [1,k-1]} \left(\sum_{\omega' \in \mathscr{L}(S')_i} \pi(\omega') \right) \left(\sum_{\omega'' \in \mathscr{L}(S'')_{k-i}} \pi(\omega'') \right) \\ &= \sum_{i \in [1,k-1]} \sum_{\substack{\omega' \in \mathscr{L}(S')_i \\ \omega'' \in \mathscr{L}(S'')_{k-i}}} \pi(\omega')\pi(\omega'') \\ &= \sum_{\omega \in \mathscr{L}(S)_k} \pi(\omega) \end{aligned}$$

- $(S \to S' \mid S'')$:

$$\mathrm{s}_k = \mathrm{s}'_k + \mathrm{s}''_k = \sum_{\omega' \in \mathscr{L}(S')_k} \pi(\omega') + \sum_{\omega'' \in \mathscr{L}(S'')_k} \pi(\omega'') = \sum_{\omega \in \mathscr{L}(S)_k} \pi(\omega)$$

Après la phase de dénombrement préliminaire, on a donc bien dans s_k la somme des poids des mots de $\mathscr{L}(S)_k$.
A partir de ce résultat, on prouve par récurrence que la phase de génération engendre bien désormais les mots du langage selon les probabilités de l'équation (4.1). Le cas terminal $(S \to t)$ est trivial, car l'équation (4.1) revient alors à $\mathbb{P}(\omega|\pi) = \frac{\pi(t)}{\pi(t)} = 1$. Dans les cas récursifs :

- $(S \to S'S'')$: Un nombre i est choisi avec probabilité $\frac{\mathrm{s}'_i \mathrm{s}''_{k-i}}{\mathrm{s}_k}$. Des mots ω' et ω'' sont alors engendrés avec probabilité $\frac{\pi(\omega')}{\mathrm{s}'_i}$ et $\frac{\pi(\omega'')}{\mathrm{s}''_{k-i}}$. Si la grammaire est non-ambiguë, alors

$$\mathbb{P}(\omega \mid \pi) = \frac{\mathrm{s}'_i \mathrm{s}''_{k-i}}{\mathrm{s}_k} \frac{\pi(\omega')}{\mathrm{s}'_i} \frac{\pi(\omega'')}{\mathrm{s}''_{k-i}} = \frac{\pi(\omega')\pi(\omega'')}{\mathrm{s}_k}$$

  où l'on retrouve bien la probabilité de l'équation (4.1) car

$$\pi(\omega) = \pi(\omega')\pi(\omega'') \quad \text{et} \quad \mathrm{s}_k = \sum_{x \in \mathscr{L}(S)_k} \pi(x).$$

- $(S \to S' \mid S'')$: Le premier tirage aléatoire choisit S' (resp. S'') avec probabilité $\frac{\mathrm{s}'_k}{\mathrm{s}_k}$ (resp. $\frac{\mathrm{s}''_k}{\mathrm{s}_k}$). L'appel récursif sur S'(resp. S'') engendre alors un mot ω avec probabilité $\frac{\pi(\omega)}{\mathrm{s}'_k}$ (resp. $\frac{\pi(\omega)}{\mathrm{s}''_k}$). La probabilité totale d'émission d'un tel mot est alors égale à

$$\mathbb{P}(\omega \mid \pi) = \frac{\mathrm{s}'_k}{\mathrm{s}_k} \frac{\pi(\omega)}{\mathrm{s}'_k} = \frac{\mathrm{s}''_k}{\mathrm{s}_k} \frac{\pi(\omega)}{\mathrm{s}''_k} = \frac{\pi(\omega)}{\mathrm{s}_k}$$

  si les langages $\mathscr{L}(S')$ et $\mathscr{L}(S'')$ sont disjoints.

Le générateur 1 modifié est donc un générateur valide pour une grammaire pondérée non-ambiguë en forme normale de Chomsky. ■

L'*impact sur la complexité binaire* de l'adjonction de pondérations se limite à une *constante* dans l'implémentation naïve. En effet, la manipulation de réels dans toute leur généralité étant inaccessible à l'informatique numérique classique[1], on se limite donc classiquement à des nombres ayant un développement binaire borné, les nombres flottants, qui équivalent à des nombres entiers multipliés par des puissances négatives de 2. En *décalant* toutes les représentations binaires de ces nombres, i.e. en les multipliant par la plus petite puissance négative de 2 présente dans l'écriture d'un poids, on obtient des nombres entiers. En remarquant que la distribution (4.1) est stable sur une multiplication des poids par un facteur $k \in \mathbb{N}$, on déduit que la seule influence des poids sur la complexité de l'algorithme par rejet réside en un facteur $\alpha = \log_2(\pi^+)$, où π^+ est le plus grand nombre représentant un poids après décalage.

4.2 Extension de la méthode de Boltzmann aux grammaires pondérées

On rappelle que le but d'un générateur aléatoire de mots d'une grammaire pondérée G est d'émettre $\omega \in \mathscr{L}(G)_n$ avec probabilité

$$\mathbb{P}(\omega \mid \pi) = \frac{\pi(\omega)}{\pi(\mathscr{L}(G)_n)}. \tag{4.2}$$

[1] En effet, pour toute séquence de bits, il existe un nombre réel qui admette cette séquence comme mantisse. Ces nombres sont tous compris entre 0 et 1, donc la théorie de l'information nous empêche de rêver d'un codage d'espérance finie ne serait ce que pour les nombre de l'intervalle $[0,1]$.

Avec C. Pivoteau et M. Soria, nous avons étudié la possibilité de transposer la méthode de Boltzman aux grammaires pondérées. On montre que cette transposition peut être réalisée au seul prix d'une substitution des séries génératrices pondérées aux séries génératrices de dénombrement.

Théorème 7 :
L'adaptation de algorithme 2, dans laquelle $S(z)$ (resp. $S'(z)$ et $S''(z)$) est remplacée par la série génératrice pondérée $S_\pi(z)$ (resp. $S'_\pi(z)$ et $S''_\pi(z)$), engendre $\omega \in \mathscr{L}(S)$ avec probabilité

$$\mathbb{P}_x(\omega \mid \pi) = \frac{\pi(\omega)x^{|\omega|}}{S_\pi(x)} \tag{4.3}$$

PREUVE : On rappelle la définition de la série génératrice pondérée $L_\pi(z)$ pour un langage $\mathscr{L}$:

$$L_\pi(z) = \sum_{\omega\in\mathscr{L}} \pi(\omega) z^{|\omega|}$$

Une fois les séries génératrices pondérées $S_\pi(z)$ substituées au $S(z)$ d'origine, on peut raisonner par induction pour obtenir les probabilités d'émission attendues.

Supposons que $\Gamma\mathscr{S}'$ (resp. $\Gamma\mathscr{S}''$) engendrent bien des mots selon une probabilité conforme à l'équation (4.3) :

- $S \to t : \mathscr{L}(S) = \{t\} \Rightarrow S_\pi(z) = \pi(t)z \Rightarrow \frac{\pi(t)x^1}{S_\pi(x)} = 1$. Or t est bien engendré avec probabilité 1.
- $S \to S' \mid S''$: On rappelle que $S_\pi(z) = S'_\pi(z) + S''_\pi(z)$ si l'union $\mathscr{L}(\mathscr{S}') \cup \mathscr{L}(\mathscr{S}'')$ est disjointe. Si $\mathscr{S}'$ est choisi, alors $\forall\omega \in \mathscr{L}(\mathscr{S}')$:

  $$\mathbb{P}_x(\omega \mid \pi) = \frac{S'_\pi(x)}{S_\pi(x)} \frac{\pi(\omega)x^{|\omega|}}{S'_\pi(x)} = \frac{\pi(\omega)x^{|\omega|}}{S_\pi(x)}$$

  Ce raisonnement vaut aussi pour un mot de $\mathscr{L}(\mathscr{S}'')$.
- $S \to S'S''$: Après avoir rappelé que $\pi(\alpha\beta) = \pi(\alpha)\pi(\beta)$ et que $S_\pi(z) = S'_\pi(z)S''_\pi(z)$ ssi il existe une unique factorisation des $\omega \in \mathscr{S}$ en un produit $\omega = \omega_1.\omega_2$, $\omega_1 \in \mathscr{S}'$ et $\omega_2 \in \mathscr{S}''$. La probabilité d'émission de ω est donc de :

  $$\mathbb{P}_x(\omega \mid \pi) = \frac{\pi(\omega_1)x^{|\omega_1|}}{S'(x)} \frac{\pi(\omega_2)x^{|\omega_2|}}{S''(x)} = \frac{\pi(\omega_1)\pi(\omega_2)x^{|\omega_1|+|\omega_2|}}{S'_\pi(x)S''_\pi(x)} = \frac{\pi(\omega)x^{|\omega|}}{S_\pi(x)}$$

L'hypothèse de correction des deux générateurs *fils* $\Gamma\mathscr{S}'$ et $\Gamma\mathscr{S}''$ est trop *forte*, et risque d'induire des *dépendances circulaires* (cycle dans le graphe de dépendance de la grammaire). On risque ainsi de supposer $\Gamma\mathscr{S}$ correct pour en déduire que $\Gamma\mathscr{S}$ est correct !
On pourra donc transposer le raisonnement par induction ci-dessus sur le nombre n d'appels à un générateur pour obtenir ω. On affaiblira alors l'hypothèse de correction en la restreignant aux objets engendrés en un nombre $\leq n$ d'appels à un générateur. Cette adaptation est essentiellement technique et omise dans un souci de simplicité. ∎

Une conséquence directe de l'équation (4.3) est le respect de la distribution induite par (4.2) au sein d'une classe de taille donnée, c'est à dire pour une génération de Boltzman en taille exacte. En taille approchée, les fréquences d'apparitions des non-terminaux sont connues pour atteindre rapidement le régime asymptotique. Elles restent donc fixes au sein des différentes classes de tailles de l'intervalle $[n(1-\varepsilon), n(1+\varepsilon)]$, et sont déterminées par les constantes c_0 de l'équation (3.12). Ce type de génération permet donc de cumuler la pondération des symboles terminaux, qui permet de jouer sur leur fréquence, à une variabilité ϵ paramètrable sur la taille des séquences générées, le tout en temps linéaire et sans précalcul ! C'est donc un outil très prometteur pour la modélisation des structures biologiques.

4.3 Grammaires pondérées et modèles markoviens

En préambule de ce chapitre consacré au produit de modèles Markoviens et grammaticaux, je tiens à remercier Michel Termier pour son opiniâtreté dans sa volonté de me voir traiter ce problème. L'idée sous-jacente à ce problème est de tenir compte à la fois de dépendances *structurelles* et *séquentielles* dans l'élaboration de modèles de couples séquence/structure pour l'ARN.

On considère des modèles structurels syntaxiques, c'est à dire des modèles restreignant l'ensemble des séquences/structures engendrées à un ensemble décrit par une grammaire. Notre but est de leur adjoindre une distribution basée sur un modèle de Markov, on réalise alors un produit de modèles, dans un sens explicité par la définition suivante.

Définition 19 (Produit d'un modèle de Markov et d'une grammaire) :
Pour tout modèle de Markov $\mathscr{M}$ et toute grammaire $\mathscr{G}$, on appelle $\mathscr{M}' = \mathscr{G} \times \mathscr{M}$ le modèle produit *de $\mathscr{M}$ et de $\mathscr{G}$ tel que, pour toute séquence $\omega \in \mathscr{L}(G)$,*

$$\mathbb{P}(\omega \mid \mathscr{M}') = \frac{\mathbb{P}(\omega \mid \mathscr{M})}{\sum_{\omega' \in \mathscr{L}(\mathscr{G})} \mathbb{P}(\omega' \mid \mathscr{M})}. \tag{4.4}$$

□

La probabilité de l'équation (4.4) est aussi celle du modèle de Markov, multipliée par l'indicatrice d'appartenance de la séquence au langage décrit par la grammaire, et normalisée pour compenser la soustraction des contributions des séquences interdites.

On prétend qu'un tel modèle peut être réalisé par une grammaire non-contextuelle pondérée. Pour cela, on proposera une preuve constructive de l'existence de la grammaire pondérée équivalente, qui constitue aussi un algorithme pour sa construction.

Théorème 8 (Markov×Grammaire⇒Grammaire pondérée) :
Pour tout modèle de Markov $\mathscr{M}$ d'ordre k et toute grammaire $\mathscr{G}$, il existe une grammaire pondérée $\mathscr{G}'$ telle pour toute séquence ω, et tout $n > k$:

$$\mathbb{P}(\omega \mid \mathscr{G} \times \mathscr{M}, n) = \mathbb{P}(\omega \mid \mathscr{G}', n)$$

PREUVE : Soit $\mathscr{M}$ un modèle de Markov d'ordre k émettant à chaque instant t un symbole α_t de Σ avec probabilités

$$\mathbb{P}(\alpha_t = x \mid \alpha_1 \dots \alpha_{t-1}) = \mathbb{P}(\alpha_t = x \mid \omega = \alpha_{t-k} \dots \alpha_{t-1}) = p_{\omega,x}.$$

On suppose sans perte de généralité la grammaire $\mathscr{G} = (\Sigma, \mathscr{N}, X, \delta)$ en forme normale de Chomsky. On propose la grammaire pondérée $\mathscr{G}' = (\Sigma', \mathscr{N}', X', \delta', \pi)$ telle que :

- Σ' est dupliqué afin d'autoriser une influence des k-lettres précédemment émises sur la lettre en cours d'émission

$$\Sigma' = \bigcup_{\omega \in \Sigma^k, t \in \Sigma} \{t_{[\omega]}\}$$

  La *sémantique* d'un symbole terminal $t_{[\omega]}$ est *symbole t émis immédiatement après la séquence* ω.

- $\mathscr{N}'$ est dupliqué pour des raisons similaires

$$\mathscr{N}' = \bigcup_{\alpha, \beta \in \Sigma^k, S \in \mathscr{N}} \{S_{\alpha \to \beta}\}$$

  La *sémantique* d'un symbole non-terminal $S_{\alpha \to \beta}$ est *non-terminal S directement précédé des symboles* α *et dont la réécriture, concaténée à* α*, admet* β *comme suffixe de longueur* k.

- X' est le nouveau symbole de départ, ou axiome.

- δ' est plus complexe, car il matérialise les dépendances. Pour tout non-terminal $S \in \mathscr{N}$ de $\mathscr{G}$ en forme normale de Chomsky :

  - $\mathscr{R}(S) = \{S \to S' \mid S''\}$:

$$\delta'_S \leftarrow \bigcup_{\alpha, \beta \in \Sigma^k} (S_{\alpha \to \beta} \to S'_{\alpha \to \beta} \mid S''_{\alpha \to \beta})$$

  - $\mathscr{R}(S) = \{S \to S'S''\}$

$$\delta'_S \leftarrow \bigcup_{\alpha, \beta, \eta \in \Sigma^k} (S_{\alpha \to \beta} \to S'_{\alpha \to \eta} S''_{\eta \to \beta})$$

  - $\mathscr{R}(S) = \{S \to t\}$

$$\delta'_S \leftarrow \bigcup_{\alpha \in \Sigma^k} (S_{\alpha \to \alpha_2 \ldots \alpha_k t} \to t_{[\alpha]})$$

  Enfin, on propose les règles $\delta'_{X'}$ suivantes pour le symbole initial X', en supposant un *état* initial ω :

$$\delta'_{X'} \leftarrow \bigcup_{\omega' \in \Sigma^k} (X \to S_{\omega \to \omega'})$$

  On propose alors $\delta' \leftarrow \bigcup_{S \in \mathscr{N}} \delta'_S \cup \delta'_{X'}$.

- Enfin, pour les pondérations π :

$$\pi_{t_{[\alpha]}} = p_{\alpha,t}, \forall\, t_{[\alpha]} \in \Sigma'$$

Lemme 1 :
Soit $\phi : \Sigma' \to \Sigma$ *un morphisme tel que*

$$\phi(t_{[\omega]}) = t$$

Alors l'extension produit/ensembliste de ϕ *est telle que pour tout* $S' \in \mathcal{N}'$, $\omega \in \Sigma^k$ *:*

$$\mathcal{L}(S) = \bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S'_{\omega \to \omega'}))$$

On peut prouver ce lemme par induction sur le nombre maximal de réécritures pour engendrer le langage associé à un non-terminal S :

- $S \to t$: Soit $\omega = x.\alpha$, on a alors les égalités

$$\bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S_{\omega \to \omega'})) = \phi(\mathcal{L}(S_{x.\alpha \to \alpha.t_{[\omega]}})) = \phi(\{t_{[\omega]}\}) = \{t\} = \mathcal{L}(S).$$

- $S \to S' \mid S''$:

$$\begin{aligned}
\bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S_{\omega \to \omega'})) &= \bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S'_{\omega \to \omega'}) \cup \mathcal{L}(S''_{\omega \to \omega'})) \\
&= \bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S'_{\omega \to \omega'})) \cup \bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S''_{\omega \to \omega'})) \\
&= \mathcal{L}(S') \cup \mathcal{L}(S'') = \mathcal{L}(S)
\end{aligned}$$

- $S \to S'S''$:

$$\begin{aligned}
\bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S_{\omega \to \omega'})) &= \bigcup_{\omega' \in \Sigma^k} \bigcup_{\omega'' \in \Sigma^k} \phi(\mathcal{L}(S'_{\omega \to \omega'})) \phi(\mathcal{L}(S''_{\omega' \to \omega''})) \\
&= \bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S'_{\omega \to \omega'})) \bigcup_{\omega'' \in \Sigma^k} \phi(\mathcal{L}(S''_{\omega' \to \omega''})) \\
&= \bigcup_{\omega' \in \Sigma^k} \phi(\mathcal{L}(S'_{\omega \to \omega'})) \mathcal{L}(S'') \\
&= \mathcal{L}(S') \mathcal{L}(S'') = \mathcal{L}(S)
\end{aligned}$$

Remarque : On n'a pas utilisé ici *explicitement* la condition de récurrence la taille maximale d'une réécriture pour un mot de taille n. Cependant celle ci est nécessaire ici, entre autre pour les cas $S \to SS'$. De plus, on remarquera que la non-ambiguïté de la grammaire interdit l'existence de *cycles* dans le graphe de dépendances entre les non-terminaux de type *union*. L'application de l'hypothèse de récurrence est donc valide.

Lemme 2 :
La probabilité d'une séquence $\omega \in \mathcal{L}(\mathcal{G}')$ *est telle que :*

$$\mathbb{P}(\omega \mid \mathcal{G}', n) = \frac{\mathbb{P}(\phi(\omega) \mid \mathcal{M})}{\sum_{\omega' \in \mathcal{L}(\mathcal{G})_n} \mathbb{P}(\omega' \mid \mathcal{M})}$$

En effet, on peut prouver par induction que les séquences de terminaux engendrés par la nouvelle grammaire sont de la forme $t^1_{\alpha_1} t^2_{\alpha_2} \dots t^n_{\alpha_n}$, où $\alpha_i = x.\beta_i \Rightarrow \alpha_{i+1} = \beta_i t^i$ et $\alpha_1 = \omega$. La probabilité d'une telle séquence est donc proportionnelle à $\Pi^n_{i\geq 0} p_{\alpha_i,t_i} = \mathbb{P}(\phi(\omega) \mid \mathscr{M})$. La partie inférieure de la fraction est juste un facteur de normalisation, donc le lemme est prouvé. ■

Le résultat précédent prouve l'inclusion de l'expressivité des modèles *produits* d'une grammaire non-contextuelle et d'un modèle de Markov dans celle des grammaires non-contextuelles pondérées. La preuve constructive de cette inclusion présentée ici peut être transposée en un algorithme pour la construction d'un tel modèle. La *taille* du modèle $\mathscr{G}'$ en nombre de non-terminaux est alors en $\mathscr{O}(|\mathscr{N}||\Sigma|^{2k})$, son nombre de règles croissant d'un facteur $\mathscr{O}(|\Sigma|^{2k})$.

On a supposé l'existence d'un état initial ω unique. Cependant, les modèles de Markov définissent traditionnellement une distribution de probabilité initiale sur les états. On pourra donc aussi prendre en compte une telle distribution, associant à chacun des *états initiaux* α une probabilité p^0_α, en *distinguant* le premier caractère émis. Pour cela, on peut construire un ensemble de règles, obtenues en *étendant* celles de $\delta'_{X'}$, et dont les premiers symboles en partant de la gauche sont systématiquement des terminaux. On remplace alors l'occurrence d'un tel terminal t_α par un non-terminal T tel que $T \rightarrow \bigcup_{\omega' \in \Sigma^k} t^0_{\alpha'}$. Les poids des nouveaux symboles terminaux seront alors tels que

$$\pi_{t^0_{[\alpha]}} = p^0_\alpha p_{\alpha,t}$$

4.4 Calcul des pondérations liées à des fréquences attendues

Dans [29], Denise et al. étudient la mise sous contrainte des fréquences des symboles terminaux dans les mots engendrés. Dans un premier temps, ils étudient la génération aléatoires de mots ayant des nombre d'occurrences imposées pour chacun des symboles terminaux. Ils proposent pour cela un algorithme, issu de l'approche récursive, en temps et en espace exponentiel sur le nombre de symboles terminaux.
Après avoir conclu à l'inefficacité de cet algorithme pour des tailles de séquences n'excédant pas quelques dizaines de bases, ils introduisent les grammaires pondérées et étudient la relation entre les paramètres du modèle pondéré et les fréquences des symboles terminaux dans les séquences émises.
Dans le chapitre qui va suivre, on va rappeler et compléter ces travaux par une optimisation de l'algorithme de génération en fréquence exacte, qui devient linéaire sur le nombre de symboles terminaux, la phase de précalcul restant exponentielle. On va ensuite rappeler les fondements théoriques de ce problème. On présentera enfin un prototype dédié à l'évaluation de pondérations réalisant des fréquences souhaitées pour les symboles terminaux.

4.4.1 Avant propos : Une optimisation de la génération en fréquence exacte

Dans un premier temps, Denise et al. étudient une transposition de l'approche récursive classique pour engendrer des mots ayant exactement $n_1, \dots, n_{|\Sigma|}$ occurrences des symboles $\alpha_1, \dots, \alpha_{|\Sigma|}$. Il dérivent un algorithme en temps $\Theta(n_1^2 n_2^2 \dots n^2_{|\Sigma|})^*$ et en espace $\Theta(n_1 n_2 \dots n_{|\Sigma|})^*$ pour la phase de dénombrement préliminaire. L'idée est de calculer l'ensemble des nombres $\mathrm{S}_{n_1,\dots,n_{|\Sigma|}}$ de mots

du langage étudié ayant n_i occurrences du symbole α_i, pour tout α_i, en utilisant une récurrence *répartissant* les $n_1, \ldots, n_{|\Sigma|}$ occurrences entre les non-terminaux :

$$(S \to \alpha_i) \quad \Rightarrow \quad \mathrm{S}_{n_1,\ldots,n_{|\Sigma|}} = \begin{cases} 1 & \text{Si } n_k = 0,\ \forall k \neq i \text{ et } n_i = 1 \\ 0 & \text{Sinon} \end{cases} \tag{4.5}$$

$$(S \to S' \mid S'') \quad \Rightarrow \quad \mathrm{S}_{n_1,\ldots,n_{|\Sigma|}} = \mathrm{S}'_{n'_1,\ldots,n'_{|\Sigma|}} + \mathrm{S}''_{n'_1,\ldots,n'_{|\Sigma|}} \tag{4.6}$$

$$(S \to S'S'') \quad \Rightarrow \quad \mathrm{S}_{n_1,\ldots,n_{|\Sigma|}} = \sum_{n'_1 \leq n_1} \cdots \sum_{n'_{|\Sigma|} \leq n_{|\Sigma|}} \mathrm{S}'_{n'_1,\ldots,n'_{|\Sigma|}} \mathrm{S}''_{n_1-n'_1,\ldots,n_{|\Sigma|}-n'_{|\Sigma|}} \tag{4.7}$$

La génération aléatoire par application de la méthode récursive est alors immédiate.

La complexité de la phase de génération, annoncée en $\Theta(n_1 n_2 \ldots n_|\Sigma|)^*$ en temps dans l'article originel, **peut être ramenée à** $\Theta(n|\Sigma|)^*$ au prix du calcul des sommes partielles pour le cas des règles de type *produit*, ce qui permettent de s'intéresser aux répartitions des n_i successivement plutôt que simultanément dans l'article.
Par exemple, il est possible de choisir une valeur pour n'_1 compatible avec l'uniformité de la génération si l'on connaît les sommes partielles $\mathrm{S}^{(k)}_{(n_2,\ldots,n_{|\Sigma|})}$, $\forall k \leq n_1$ telles que

$$\mathrm{S}^{(k)}_{(n_2,\ldots,n_{|\Sigma|})} = \sum_{n'_2 \leq n_2} \cdots \sum_{n'_{|\Sigma|} \leq n_{|\Sigma|}} \mathrm{S}'_{k,\ldots,n'_{|\Sigma|}} \mathrm{S}''_{n_1-k,\ldots,n_{|\Sigma|}-n'_{|\Sigma|}}.$$

La probabilité d'une valeur candidate n'_1 est alors donnée par $\mathbb{P}(n'_1 \mid \varnothing) = \frac{\mathrm{S}^{(n'_1)}_{(n_2,\ldots,n_{|\Sigma|})}}{\mathrm{S}_{n_1,\ldots,n_{|\Sigma|}}}$.
Dans le cas général, une fois des valeurs déterminées pour $n'_1, \ldots, n'_{i-1}$, on *prolonge l'affectation* à n'_i en utilisant les sommes partielles

$$\mathrm{S}^{(n'_1,\ldots,n'_{i-1},k)}_{(n_{i+1},\ldots,n_{|\Sigma|})} = \sum_{n'_{i+1} \leq n_{i+1}} \cdots \sum_{n'_{|\Sigma|} \leq n_{|\Sigma|}} \mathrm{S}'_{n'_1,\ldots,n'_{i-1},k,n'_{i+1}\ldots,n'_{|\Sigma|}} \mathrm{S}''_{n_1-k,\ldots,n_{|\Sigma|}-n'_{|\Sigma|}}.$$

qui permettent de choisir une valeur n'_i avec probabilité $\mathbb{P}(n'_i \mid n'_1, \ldots, n'_{i-1}) = \frac{\mathrm{S}^{(n'_1,\ldots,n'_i)}_{(n_{i+1},\ldots,n_{|\Sigma|})}}{\mathrm{S}^{(n'_1,\ldots,n'_{i-1})}_{(n_i,\ldots,n_{|\Sigma|})}}$. Une preuve rapide de la correction de cette optimisation peut être fournie par le calcul de la probabilité d'une répartition $(n'_1, \ldots, n'_{|\Sigma|})$:

$$\begin{aligned} \mathbb{P}(n'_1,\ldots,n'_{|\Sigma|}) &= \mathbb{P}(n'_1 \mid \varnothing)\mathbb{P}(n'_2 \mid n'_1)\ldots\mathbb{P}(n'_{|\Sigma|} \mid n'_1,\ldots,n'_{|\Sigma|-1}) \\ &= \frac{\mathrm{S}^{(n'_1)}_{(n_2,\ldots,n_{|\Sigma|})}}{\mathrm{S}_{n_1,\ldots,n_{|\Sigma|}}} \frac{\mathrm{S}^{(n'1,n'_2)}_{(n_3,\ldots,n_{|\Sigma|})}}{\mathrm{S}^{(n'_1)}_{n_2,\ldots,n_{|\Sigma|}}} \cdots \frac{\mathrm{S}^{(n'_1,\ldots,n'_{|\Sigma|})}_{(\varnothing)}}{\mathrm{S}^{(n'_1,\ldots,n'_{|\Sigma|-1})}_{(n_{|\Sigma|})}} \\ &= \frac{\mathrm{S}^{(n'_1,\ldots,n'_{|\Sigma|})}}{\mathrm{S}_{n_1,\ldots,n_{|\Sigma|}}} = \frac{\mathrm{S}'_{n'_1,\ldots,n'_{|\Sigma|}} \mathrm{S}''_{n_1-n'_1,\ldots,n_{|\Sigma|}-n'_{|\Sigma|}}}{\mathrm{S}_{n_1,\ldots,n_{|\Sigma|}}} \end{aligned}$$

On retrouve bien ici la contribution de la répartition $(n'_1, \ldots, n'_{|\Sigma|})$ à la somme de l'équation (4.7).
Le calcul des sommes partielles ne modifie pas la complexité en temps de la phase de précalcul,

car il peut être réalisé *à la volée* au cours du calcul des $\mathrm{s}_{n_1,\dots,n_{|\Sigma|}}$. Cependant, il introduit un facteur $|\Sigma|$ supplémentaire dans la complexité en mémoire, car l'espace des $\mathrm{s}^{(n'_1,\dots,n'_{i-1},k)}_{(n_{i+1},\dots,n_{|\Sigma|})}$ à calculer est égal à celui des $\mathrm{s}_{n_1,\dots,n_{|\Sigma|}}$, pour chaque $i \in [1,|\Sigma|]$. En combinant cette optimisation avec une stratégie *boustrophédon* (voir section 3.3.3), on obtient la complexité annoncée.

4.4.2 Evaluation de la fréquence d'un symbole dans une grammaire pondérée

Le principal problème qui nous intéresse est le suivant : Comment calculer une pondération π telle que les symboles terminaux apparaissent avec des fréquences imposées *a priori* ? Cette interrogation nous amène à nous intéresser tout d'abord au problème inverse, c'est à dire à l'évaluation des fréquences des terminaux dans une grammaire dont la pondération est connue. En effet, si cette relation s'avère très simple, il est alors possible de l'inverser. Plus généralement, la capacité à évaluer de telles fréquences permet la mise en oeuvre d'approches heuristiques issues du domaine de l'optimisation, qui pourraient même se révéler déterministes si certaines conditions sont remplies.

On commence par définir formellement la fréquence d'apparition d'un symbole terminal dans une grammaire pondérée.

Définition 20 (Fréquence d'apparition d'un symbole terminal) :
On appelle fréquence d'apparition d'un symbole α_i *dans une grammaire pondérée* $\mathscr{G}_\pi$ *pour une taille* n *la valeur réelle* $\mu_{\alpha_i,n} \in [0,1]$ *telle que*

$$\mu_{\alpha_i,n} = \frac{\mathbb{E}(X_n = |\omega|_{\alpha_i} \mid \pi, |\omega| = n)}{n}$$

où X_n *est la variable aléatoire associée au nombre d'occurrences de* α_i *dans un mot* ω *engendré selon la probabilité définie par* π. □

De plus, on notera μ_{α_i} l'asymptotique de $\mu_{\alpha_i,n}$:

$$\mu_{\alpha_i} = \lim_{n\to\infty} \mu_{\alpha_i,n}$$

Evaluation symbolique

Il est possible de calculer les fréquences $\mu_{\alpha_i,n}$ à partir de la série génératrice $G_\pi(z,u_1,\dots,u_i,\dots,u_{|\Sigma|})$ multivariée en les différents symboles terminaux. En effet

$$\begin{aligned}
G_\pi(z,1,\dots,u_i,\dots,1) &= \sum_{\omega\in\mathscr{L}(\mathscr{G}_\pi)} \pi(\omega)\, u_i^{|\omega|_{\alpha_i}} z^{|\omega|} \\
\frac{\partial G_\pi(z,1,\dots,u_i,\dots,1)}{\partial u_i}(z,1) &= \sum_{\omega\in\mathscr{L}(\mathscr{G}_\pi)} \pi(\omega)|\omega|_{\alpha_i} z^{|\omega|} \\
\frac{[z^n]\frac{\partial G_\pi(z,1,\dots,u_i,\dots,1)}{\partial u_i}(z,1)}{n[z^n]G_\pi(z,1,\dots,1)} &= \frac{\sum_{\omega\in\mathscr{L}(\mathscr{G}_\pi)_n} \pi(\omega)|\omega|_{\alpha_i}}{n\sum_{\omega\in\mathscr{L}(\mathscr{G}_\pi)_n}\pi(\omega)} = \frac{\sum_{\omega\in\mathscr{L}(\mathscr{G}_\pi)_n}\mathbb{P}(\omega\mid\pi)|\omega|_{\alpha_i}}{n} \\
&= \frac{\mathbb{E}(X_n = |\omega|_{\alpha_i}\mid\pi)}{n} = \mu_{\alpha_i,n}
\end{aligned}$$

De plus, l'extraction des deux termes de degré n peut être accélérée par l'utilisation de récurrences linéaires, dérivables automatiquement par le biais de `GFun` [83], en remarquant que la série génératrice numérateur $\frac{\partial G_\pi(z,1,\ldots,u_i,\ldots,1)}{\partial u_i}(z,1)$ est algébrique, car la série génératrice d'un langage obtenu à partir de $\mathscr{L}(G_\pi)$ en *marquant* une occurrence de α_i. On trouvera de plus amples détails sur l'opérateur de pointage et son impact sur la série génératrice et sur la grammaire engendrant le langage dans [36]. Le calcul de ces fréquences peut donc être obtenu en temps $\mathscr{O}(n^2)$ à partir de la série génératrice bivariée.

Une autre approche utilisant le calcul symbolique consiste à évaluer l'asymptotique des dérivées partielles, qui sont aussi des séries génératrices algébriques[2], et dont on sait donc extraire automatiquement un équivalent asymptotique pour le n-ième coefficient. On peut pour cela encore une fois utiliser `GFun` [83] pour dériver automatiquement des équivalents asymptotiques des $[z^n]\frac{\partial G_\pi(z,1,\ldots,u_i,\ldots,1)}{\partial u_i}(z,1)$ et de $[z^n]G_\pi(z,1,\ldots,1)$.

Enfin, il est possible d'appliquer la variante du théorème de Drmota présenté en section 4.4.3 pour ce problème.

Approche récursive

Les deux premières approches présentées ci-dessus présentent l'avantage d'être de faibles complexités. Cependant, elles supposent les séries génératrices de langage déjà acquises, ce qui n'est pas toujours tout à fait réaliste, quand les grammaires sont complexes. La troisième nécessite la résolution d'un système d'équations complexe, et requiert que la grammaire respecte de nombreuses propriétés non-triviales.

On propose ici une approche complémentaire, adaptée de la phase de dénombrement préliminaire de l'approche récursive.

On s'intéresse au nombre moyen $f_{\alpha_i,S,n}$ d'occurrences d'une lettre α_i au sein d'un mot de taille n, issu de S et engendré selon la distribution induite par π :

$$\begin{aligned}
f_{\alpha_i,S,n} &= \mathbb{E}(|\omega|_{\alpha_i} \mid \pi, n) \\
&= \sum_{\omega\in\mathscr{L}(S)_n} \mathbb{P}(\omega \mid \pi)|\omega|_{\alpha_i} \\
&= \sum_{\omega\in\mathscr{L}(S)_n} \frac{\pi(\omega)}{\pi(\mathscr{L}(S)_n)}|\omega|_{\alpha_i} \\
&= \frac{1}{\pi(\mathscr{L}(S)_n)} \sum_{k=0}^{n} k \sum_{\substack{\omega\in\mathscr{L}(S)_n \\ |\omega|_{\alpha_i}=k}} \pi(\omega) \\
&= \frac{\sum_{k=0}^{n} k\sigma_{\alpha_i,S,n,k}}{\pi(\mathscr{L}(S)_n)}
\end{aligned}$$

[2] Le *marquage* d'une occurrence de α_i équivaut dans le domaine des séries génératrices multivariées à une dérivée partielle et réciproquement.

où $\sigma_{\alpha_i,S,n,k} = \sum_{\substack{\omega\in\mathscr{L}(S)_n \\ |\omega|_{\alpha_i}=k}} \pi(\omega)$. On peut aussi dire que $\sigma_{\alpha_i,S,n,k}$ *est la somme des poids des mots de taille n issus de S ayant k occurences de la lettre* α_i. On remarque que la connaissance des $\sigma_{\alpha_i,S,k,l}$ nous conduit en temps $\mathscr{O}(n)$ à celle des $f_{\alpha_i,S,k}$ car $\pi(\mathscr{L}(S)_n) = \sum_{k=0}^{n} \sigma_{\alpha_i,S,n,k},\ \forall \alpha_i \in \Sigma$.

On propose la récurrence suivante pour le calcul des $\sigma_{\alpha_i,S,n,k}$, en supposant la grammaire en forme normale de Chomsky sans production ε :

$$
\begin{aligned}
S \to \alpha_j \quad &\Rightarrow \quad \sigma_{\alpha_i,S,n,k} = \begin{cases} \pi(\alpha_i) & \text{Si } \alpha_i = \alpha_j,\ n = 1 \text{ et } k = 1 \\ \pi(\alpha_j) & \text{Si } \alpha_i \neq \alpha_j,\ n = 1 \text{ et } k = 0 \\ 0 & \text{Sinon} \end{cases} \\
S \to S \mid S'' \quad &\Rightarrow \quad \sigma_{\alpha_i,S,n,k} = \sigma_{\alpha_i,S',n,k} + \sigma_{\alpha_i,S'',n,k} \\
S \to S'S'' \quad &\Rightarrow \quad \sigma_{\alpha_i,S,n,k} = \sum_{a=1}^{n-1} \sum_{b=0}^{k} \sigma_{\alpha_i,S',a,b} \cdot \sigma_{\alpha_i,S'',n-a,k-b}
\end{aligned}
$$

Une fois ces nombres calculés, on obtient les fréquences $\mu_{\alpha_i,n}$ en remarquant que

$$\mu_{\alpha_i,n} = f_{\alpha_i,X,n} = \frac{\sum_{k=0}^{n} k\sigma_{\alpha_i,X,n,k}}{\sum_{k=0}^{n} \sigma_{\alpha_i,X,n,k}}$$

où X est l'axiome de la grammaire.

Le stockage des $\sigma_{\alpha_i,S,n,k}$ nécessite un tableau de dimension $\mathscr{O}(n^2)$, où n est la taille des séquences à engendrer. Le calcul d'une de ces $\mathscr{O}(n^2)$ entrées nécessite au pire $\mathscr{O}(n^2)$ opérations arithmétiques pour l'opération produit. Le calcul des fréquences exactes est donc en $\mathscr{O}(n^4)^*$.

4.4.3 Calcul des pondérations

Approche analytique

Un résultat dû à Drmota [34], transposé par Denise et al dans [29] permet d'envisager une évaluation *statique* des pondérations réalisant des fréquences attendues.

Définition 21 (Grammaire de type simple) :
Soit $\mathscr{G} = (\Sigma, \mathscr{N}, X, \delta)$ *une grammaire non-contextuelle et* $\mathrm{s}_{n,k_1,\ldots,k_{|\Sigma|-1}}$ *le nombre de mots issus du non-terminal X de taille n et ayant* k_j *occurrences de la lettre* α_j, $j \in [1, |\Sigma| - 1]$.
$\mathscr{G}$ *est alors de type simple si il existe* $|\mathscr{N}|$ *cônes* $|\Sigma|$*-dimensionnels* $\mathscr{C}_i \subset \mathbb{R}^{|\Sigma|}$ *associés à des non-terminaux* S_i*, centrés en* 0*, et* saturés*, c'est à dire que pour tout* $(n, k_1, \ldots, k_{|\Sigma|}) \in \mathscr{C}_i \cap \mathbb{N}^{|\Sigma|}$*, on a* $\mathrm{s}_{i\ n,k_1,\ldots,k_{|\Sigma|-1}} \neq 0$. □

Cette notion garantit l'existence d'un *noyau dense* dans l'espace des $\mathrm{s}_{i\ n,k_1,\ldots,k_{|\Sigma|}}$, qui contraint asymptotiquement les nombres d'occurrences de chaque α_j dans un mot du langage $\mathscr{L}(S_i)$ à un comportement en $\frac{\kappa_{i,j}}{n}$.
On rappelle que la série génératrice multivariée de langage $S(x, u_1, \ldots, u_{|\Sigma|})$ associée à un non-terminal S est définie telle que

$$S(x, u_1, \ldots, u_{|\Sigma|}) = \sum_{\omega\in\mathscr{L}(S)} x^{|\omega|} u_1^{|\omega|_{\alpha_1}} \ldots u_{|\Sigma|}^{|\omega|_{\alpha_{|\Sigma|}}} = \sum_{n\geq 0} \sum_{\substack{k_1\geq 0 \\ \ldots \\ k_{|\Sigma|}\geq 0}} \mathrm{s}_{n,k_1,\ldots,k_{|\Sigma|}} x^n u_1^{k_1} \ldots u_{|\Sigma|}^{k_{|\Sigma|}}$$

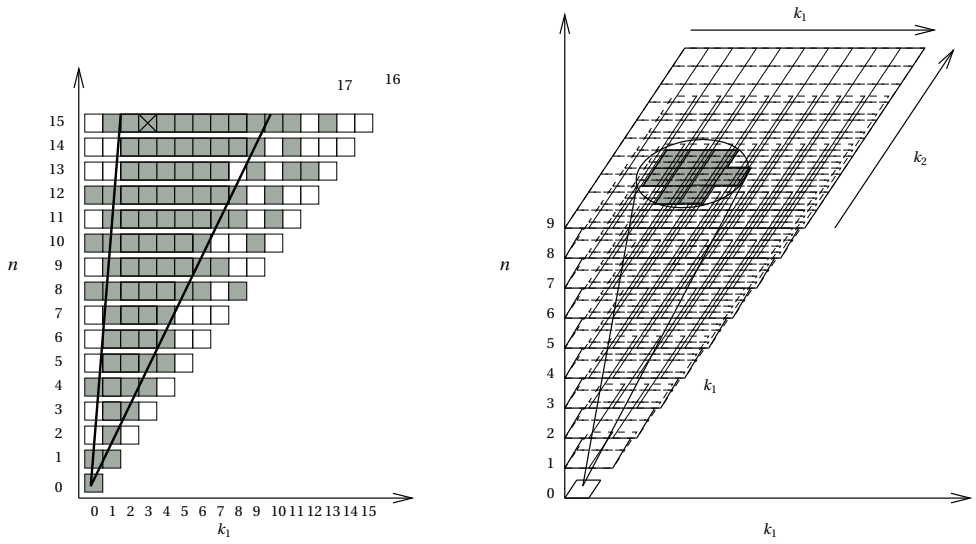


Fig. 4.1: Exemples de cône Σ-dimensionnels saturés par des langages à deux (gauche) et trois (droite) symboles terminaux. Les cases grisées représentent des coefficients saturés. Par exemple, la case $(4,15)$ (croix) est grisée dans la partie gauche, ce qui signifie qu'il existe au moins un mot issu de la grammaire, de taille 15 et faisant apparaître 4 occurrences du symbole α_1.

Exemple :
Reprenons le langage des palindromes décrit par une grammaire ayant un axiome S et une règle $S \rightarrow aSa \mid bSb \mid varepsilon$. Il est possible de transposer les règles d'une grammaire en équations sur les séries génératrices multivariées manipulées en *marquant* tous les terminaux avec la variable x et celles de a (resp. b) avec la variable u_a (resp. u_b).

La série génératrice multivariée $S(x,u_a,u_b)$ est alors telle que :

$$S(x,u_a,u_b) = xu_aS(x,u_a,u_b)xu_a + xu_bS(x,u_a,u_b)xu_b + 1$$

Si l'on ajoute une pondération π à la grammaire, alors la récurrence ci dessus devient

$$S_\pi(x,u_a,u_b) = \pi_a xu_aS_\pi(x,u_a,u_b)\pi_a xu_a + \pi_b xu_bS_\pi(x,u_a,u_b)\pi_b xu_b + 1$$

Théorème 9 (Asymptotique des fréquences des terminaux [34; 29]) :
Soit $\mathscr{G}_\pi = (\Sigma, \mathscr{N}, X, \delta, \pi)$ une grammaire non contextuelle telle que :

1. $\mathscr{L}(\mathscr{G}_\pi) \notin Rat$.
2. $\mathscr{G}_\pi$ *n'admet pas de production* ε.
3. $\mathscr{G}_\pi$ *est de* type simple.

4. *$\mathscr{G}_\pi$ est fortement connexe (voir définition 17 en page 45).*

Soient S_j les séries génératrices associées aux non-terminaux, et $S_i = \Phi_i(x, u_1, \ldots, u_{|\Sigma|}, S_1, \ldots, S_{|\mathscr{N}|})$ les relations déduites des règles de la grammaire.
Soit A la matrice des dérivées partielles de Φ telle que $A = \left(\frac{\partial \Phi_i}{\partial S_j}\right), i, j \in [1, |\mathscr{N}|]$. On considère le système suivant :

$$\begin{cases} S_1(x\pi_1 u_1, \ldots, x\pi_{|\Sigma|} u_{|\Sigma|}) = \Phi_1(x, u_1, \ldots, u_{|\Sigma|}, S_1, \ldots, S_{|\mathscr{N}|}) \\ \ldots \\ S_{|\mathscr{N}|}(x\pi_1 u_1, \ldots, x\pi_{|\Sigma|} u_{|\Sigma|}) = \Phi_{|\mathscr{N}|}(x, u_1, \ldots, u_{|\Sigma|}, S_1, \ldots, S_{|\mathscr{N}|}) \\ 0 = \det \mathbb{I} - A \end{cases} \tag{4.8}$$

□

Soit $(x, S_1, \ldots, S_{|\mathscr{N}|})$ un $|\mathscr{N}|+1$-uplet de fonctions en les variables $u_1, \ldots, u_{|\Sigma|}$, solution du système (4.8) et tel que $x(1, \ldots, 1) > 0$ Alors :

$$\mu_{\alpha_i} = \frac{1}{x(1, \ldots, 1)} \frac{\partial x}{\partial u_i}(1, \ldots, 1)$$

Remarque :
On peut relâcher la contrainte 2 car la mise en forme normale de Chomsky fait remonter les ε vers un axiome absent des parties droites des règles. Il suffit alors de supprimer cet axiome *artificiel*, et d'utiliser comme nouvel axiome sa partie droite. Le langage obtenu est alors le même que celui de la grammaire originale, à ε près. De plus, en substituant leurs parties droites aux occurrences des non-terminaux n'engendrant qu'un langage fini, on obtient une grammaire fortement connexe si celle d'origine l'était aussi.

On peut donc, en introduisant des poids π dans les polynômes Φ_i, obtenir une relation, parfois étonnamment simple, entre les poids et les fréquences des non-terminaux, comme en atteste l'application à l'ARN de la section 7.3.2, page 107.
Cependant la résolution du système (4.8) dans le cas de pondérations multiples reste encore problématique, car inaccessible à un logiciel de calcul scientifique, et les conditions d'application du théorème sont un peu restrictives. Nous avons donc développé une approche *orthogonale* basée sur l'optimisation de fonction.

Approche optimisation

Comme nous l'avons vu dans la section 4.4.2, nous disposons de plusieurs méthodes pour évaluer les fréquences μ_{α_i} des différents terminaux dans une grammaire pondérée par une fonction π. On peut donc, en comparant ces fréquences à celle souhaitées f_{α_i}, associer algorithmiquement à chaque fonction π candidate un score basé sur la distance aux fréquences souhaitées. On dispose alors de ce que les membres de la communauté *optimisation* qualifient de *fonction objectif*, c'est à dire d'une fonction à k paramètres pour laquelle on souhaite trouver un maximum. De nombreuses approches ont alors été développées, suivant que l'on dispose ou non d'une forme close pour la fonction objectif, de ses dérivées, de son comportement au voisinage

d'un point ou encore d'une propriété d'unicité du minimum. On peut aussi exiger une convergence ou se contenter d'une approximation obtenue rapidement, et risquant de *stagner* par la suite.

Dans notre cas, on était intéressé par la construction d'une fonction de pondération π réalisant des fréquences attendues f_i ou, à défaut, s'en approchant le plus possible. Une approche analytique pouvant traiter les cas *simples*, c'est à dire les cas où les grammaires proposées ont peu de symboles non-terminaux et des règles peu autoréférentielles, on réservera les méthodes issues de l'optimisation à des cas plus complexes, en utilisant pour la fonction objectif l'évaluation *algorithmique* des fréquences présentée en section 4.4.2.

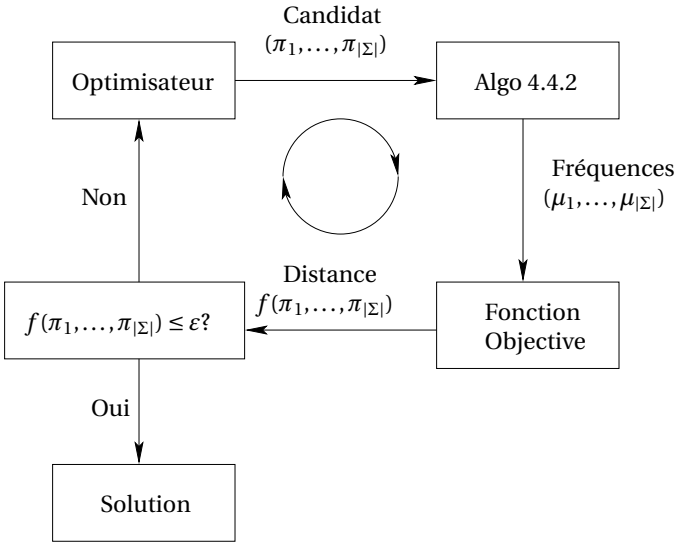


Fig. 4.2: Schéma de fonctionnement de notre optimisateur pour le calcul de la pondération π réalisant des fréquences f_i pour les terminaux α_i.

Il existe enfin deux grandes catégories d'optimisateurs :

- L'approche *linéaire*, qui chemine dans l'espace des paramètres en se laissant *couler* le long des pentes les plus fortes. Dans le cas d'une expression explicite de la fonction objectif, le plus célèbre représentant de cette famille est la *descente de gradients* bien connue des physiciens.
- L'approche *intervalle de confiance*, qui approche le comportement local en un point de la fonction objectif par un polynôme. Elle alterne des étapes de raffinement de l'approximation, permettant d'en étendre le rayon de pertinence, et des étapes de translation vers le maximum.

Nous avons donc couplé une implémentation de l'algorithme de la section 4.4.2 à deux implémentations d'optimisateurs dues à F. VanDen Berghen : L'algorithme de Rosenbrock [82] et celui

de Vanden Berghen [14]. Nous avons pour cela utilisé comme fonction objectif des fonctions f_e et f_r telles que :

$$f_e(\pi_1,\ldots,\pi_{|\Sigma|}) = \sqrt{\sum_{i=1}^{|\Sigma|}(f_i-\mu_i)^2} \quad f_r(\pi_1,\ldots,\pi_{|\Sigma|}) = \sqrt{\sum_{i=1}^{|\Sigma|}(\frac{f_i-\mu_i}{f_i})^2}$$

La fonction f_e est une distance euclidienne classique, et la fonction f_r est une distance après normalisation. Cette dernière a pour objectif d'attirer l'attention de l'optimisateur sur les petites fréquences, qui avaient tendance à être négligée, car peu contributive à f_e.

Notre idée était de comparer les performance de représentants des deux familles d'optimisateurs. La comparaison de ces deux optimisateurs sur des exemples réels, où l'évaluation de la fonction objectif est très coûteuse ($\mathcal{O}(n^5)$), s'est révélée nettement favorable à l'algorithme de Van-Den Berghen, qui semble converger *logarithmiquement* sur les cas testés. Par logarithmique, on entend que la distance aux fréquences attendues décroît exponentiellement sur le nombre d'évaluation de la fonction objectif. Cependant, nous avons conservé l'algorithme de Rosenbrock dans le logiciel, car celui ci fait moins d'hypothèse sur le comportement de la fonction à optimiser. En particulier, l'approximabilité des comportements locaux des fonctions f_e et f_r par des polynômes de degré 2 nécessite une étude spécifique sur le plan théorique, qui n'a pas été fournie pour l'instant et figure dans les perspective de la présente thèse. L'algorithme de Rosenbrock est quant à lui beaucoup plus permissif sur ce sujet, tolérant par exemple des fonctions objectifs non-dérivables. Des applications de ce logiciel à la génération de structures d'ARN réalistes sont présentées en section 7.

5. GENRGENS : GÉNÉRATION DE SÉQUENCES ET STRUCTURES ALÉATOIRES

5.1 Motivation

Avec mes coauteurs à l'origine de GenRGenS, à savoir Michel Termier et Alain Denise, nous avons oeuvré à sensibiliser la communauté bio-informatique à l'intérêt de modèles de séquences et structures aléatoires. Le constat de départ étant qu'il n'existait pas ou peu d'outils permettant la génération aléatoire de séquences biologiques structurées, ou combinant différents formalismes. Une illustration de la nécessaire prise en charges de modèles plus complexes que les chaînes de Markov phénomène peut être trouvée dans la partie 9 et plus particulièrement dans [53], où nous avons prouvé l'existence d'un biais dans la mesure de sensibilité d'un algorithme, mesure basée sur une modélisation par chaîne de Markov là où les objets considérés étaient en réalité structurés et complexes. Concrètement, cette sensibilisation a donné lieu à une publication de GenRGenS dans le journal *Bioinformatics* [76], à sa présentation en de nombreuses conférences [72; 28; 75], à l'élaboration d'un manuel introductif très complet [74] et à l'élaboration d'un site internet dédié à la cause et accessible à l'adresse :

http://www.lri.fr/bio/GenRGenS

GenRGenS suscite un intérêt croissant au sein de la communauté, il a été téléchargé à partir de 216 adresses IP distinctes originaires de 24 pays sur la seule période Décembre 2005-Août 2006. Il apparaît dans de nombreuses thèses [80; 50; 13; 2] récentes en Bioinformatique et articles de recherche [7; 93; 94][1].

5.2 Fonctionnement général

GenRGenS est un logiciel *boîte à outil* dédié à la génération aléatoire de séquences génomiques. Il a pour vocation de permettre à son utilisateur d'engendrer des séquences aléatoires à partir d'un modèle statistique/syntaxique sans avoir à se soucier des détails algorithmiques. L'utilisateur formule le modèle dans un fichier de description en utilisant pour cela un langage *de haut niveau*. Il fournit alors la taille de base n des séquences et leur nombre k, et GenRGenS procède à la génération aléatoire de ces séquences, les renvoyant sur la sortie standard. Au sein de GenRGenS, une abstraction de la notion de générateur aléatoire a été mis en place afin de permettre de futures extensions à de nouveaux modèles de séquences. L'interface Generator implémente cette abstraction, selon le schéma de la figure 5.1.

[1] Il est intéressant de noter que les auteurs de [93; 94] ont utilisé GenRGenS pour engendrer des exemples positifs pour une compétition d'algorithmes d'inférence grammaticale, ce qui représente une application nouvelle et prometteuse des principes de génération aléatoire à base de grammaire non-contextuelles.

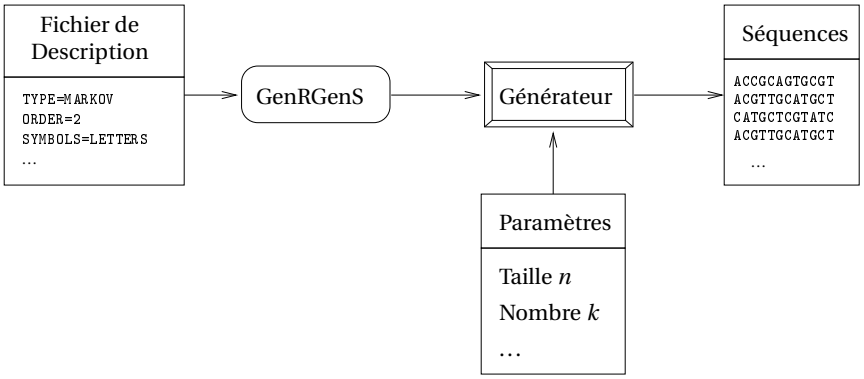


Fig. 5.1: Fonctionnement général de `GenRGenS` : Un générateur, implémentant l'interface `Generator`, est créé à partir d'un fichier décrivant le modèle de séquences aléatoires. Il est par la suite utilisé pour engendrer k séquences de taille n.

5.3 Modèles pris en charge

`GenRGenS` ayant pour vocation d'être utilisé par des non-informaticiens capable de formuler un modèle de séquences aléatoires, il comprend des générateurs pour les principales classes de modèles utilisés en bioinformatique.

5.3.1 Modèles de Markov

Définition statistique

Les modèles de Markov pour les séquences s'attachent à capturer des dépendances de courtes portées entre les symboles. Formellement, soient $X_1, \ldots, X_n$ les variables aléatoires dont les contenus sont les symboles de la séquence, on dit d'un modèle M qu'il est markovien si le contenu de la $n+1$-ième lettre ne dépend que des k lettres précédemment émise :

$$\mathbb{P}(X_{n+1} = \alpha \mid M, X_n, \ldots, X_{n-k}) = \mathbb{P}(X_{n+1} = \alpha \mid M, X_n, \ldots, X_{n-k-i});\ \forall i \in [0, n-k+1]$$

Le paramètre k du modèle est alors appelé l'*ordre*. On parle alors de *processus à mémoire finie*. Un modèle de Markov est donc caractérisé par la donnée de son ordre k et celle de toutes les probabilités $\mathbb{P}(X_{n+1} = \alpha \mid M, X_n, \ldots, X_{n-k})$, aussi appelées *probabilités de transition*. On appelle *contexte* le *suffixe* $\omega_{n,k} := X_n, \ldots, X_{n-k}$ composé des valeurs des k derniers symboles émis.

`BuildMarkov` : Apprendre un modèle de Markov d'un ensemble de séquences

Un modèle de Markov standard peut être appris à partir d'un ensemble de séquences en une seule passe sur l'ensemble des séquences. De plus, la mise en forme du fichier de description pour un *grand* modèle[2] peut s'avérer particulièrement fastidieux. C'est pourquoi nous

[2] Le nombre de probabilités de transitions à fournir croît en $\Theta(|\Sigma|^k)$,

avons développé un outil `BuildMarkov` qui construit le modèle de Markov associé à l'ensemble de séquences, puis le renvoie sous la forme d'un fichier de description *prêt à l'emploi* dans `GenRGenS`.

Variantes du modèle standard

Au cours du développement de `GenRGenS`, il est vite apparu que, outre le modèle standard ou *homogène*, deux variantes du modèle markovien étaient utilisées par la communauté :

- Les modèles de Markov hétérogènes :
  Il se caractérisent par la prise en compte d'un paramètre *nombre de phases* p supplémentaire. Ainsi, les probabilités d'émission pour un symbole X_n tiendront compte, outre les k derniers symboles émis, de la position $(n \bmod p)$ de X_n relativement à une *fenêtre* de taille p. Les nouvelles probabilités d'émission pour un modèle M' sont donc données par $\mathbb{P}(X_{n+1} = \alpha \mid M', \omega_{n,k}, n \bmod p)$.

  Cette adjonction permet de simuler à faible coût l'importance de la position d'une base au sein d'un codon, dans le cas de l'ARN messager et des zones de l'ADN codant pour des gènes, comme l'illustre la figure 5.2.

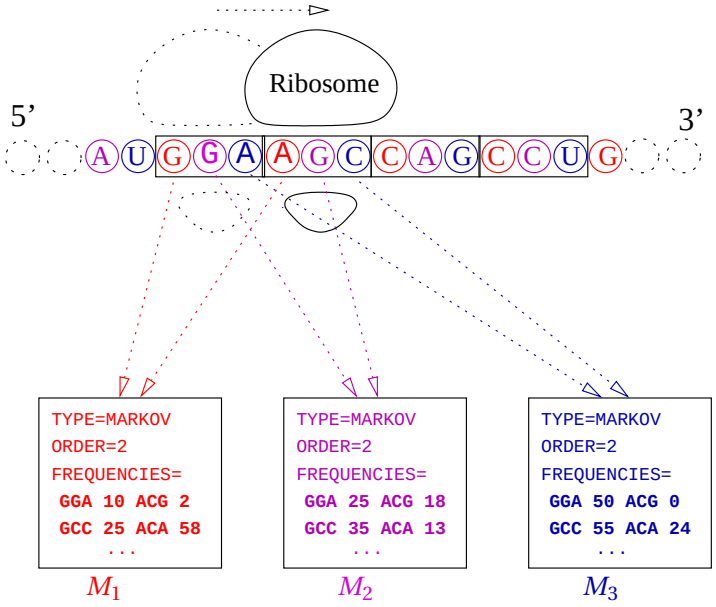


Fig. 5.2: Lors de la transcription de l'ARN, les bases constituant les codons ne jouent pas des rôles identiques, on doit alors utiliser des modèles différents (M_1, M_2 et M_3) pour les différentes *phases*.

Cependant, il ne s'agit pas d'un enrichissement du modèle de Markov, car il est possible d'opérer p copies disjointes $\Sigma_0, \ldots, \Sigma_{p-1}$ du vocabulaire Σ initial, et de fixer dans ce modèle

de Markov homogène M'' les probabilités de transitions suivantes :

$$\mathbb{P}(X_{n+1} = \alpha \mid M'', \omega_{n,k}) = \begin{cases} \mathbb{P}(X_{n+1} = \alpha \mid M', \omega_{n,k}, i) & \text{Si } \alpha \in \Sigma_i, X_n \in \Sigma_j \\ & \text{et } i = (j+1) \bmod p \\ 0 & \text{Sinon} \end{cases}$$

La probabilité d'une séquence dans M'', dans laquelle on *ramène* les caractères émis au vocabulaire Σ initial, est alors la même que dans le modèle M'.

- Les modèles de Markov cachés (HMM) :
  Également appelés modèles de Markov *multi-urnes*, ils permettent de modéliser la décomposabilité des séquences génomiques en régions gouvernées par des règles spécifiques. Par exemple, dans l'ADN, il est possible de discriminer partiellement entre les zones codant pour des protéines et celles intergéniques (non-codantes) uniquement en observant les proportions de mots de k lettres dans ces régions. Transposé dans le domaine markovien, la modélisation d'un tel phénomène implique l'utilisation de deux niveaux d'abstraction :

  - Un premier modèle M_h va modéliser l'alternance des régions codantes et non-codantes. Ces états sont qualifiés de *cachés*, d'où le nom donné au formalisme.
  - Des modèles de Markov classiques M_i, M_{c_1}, M_{c_2} et M_{c_3} se chargeront des compositions en bases des régions intergéniques (Modèle M_i) ou codantes (M_{c_1},M_{c_2} et M_{c_3} modélisant les trois positions des bases dans le codon).

  Il en résulte le modèle de Markov caché de la figure 5.3, pour lesquels les probabilités du modèle M_i ne sont pas instanciées, et peuvent être apprises sur les séquences réelles. L'adjectif *caché* vient du problème historique ayant vu l'apparition de ce formalisme, à savoir : *Etant donnée une séquence (ADN, par exemple), et un modèle de Markov caché, trouver la séquence d'états cachés (Régions codantes/non-codantes) qui maximise la vraisemblance*. En effet, seule la séquence est disponible, et les états dans lesquels ont été émis ses symboles sont *cachés*, toute la difficulté consistant à les révéler.

  Encore une fois, on peut prouver qu'en construisant des copies $\Sigma_{\alpha\beta}$ du vocabulaire, pour tout α et β états du modèle caché, et en prenant pour probabilité d'émission de $s_{\alpha\beta}$ le produit de la probabilité d'émission d'un symbole s dans α et de probabilité de transition $\beta \to \alpha$, alors on obtient un modèle de Markov classique présentant des mêmes probabilité d'émission qu'un modèle caché donné. L'expressivité des modèles de Markov cachés n'excède donc pas celle du modèle classique.

Ces deux variantes du modèle classique sont prises en charge par `GenRGenS`.

5.3.2 Expressions régulières, motifs PROSITE

Présentation générale

Selon la distinction introduite en section 3.1.2, et par opposition au modèles de Markov, les modèles de séquences basés sur des expressions régulières rentrent dans la catégorie des modèles

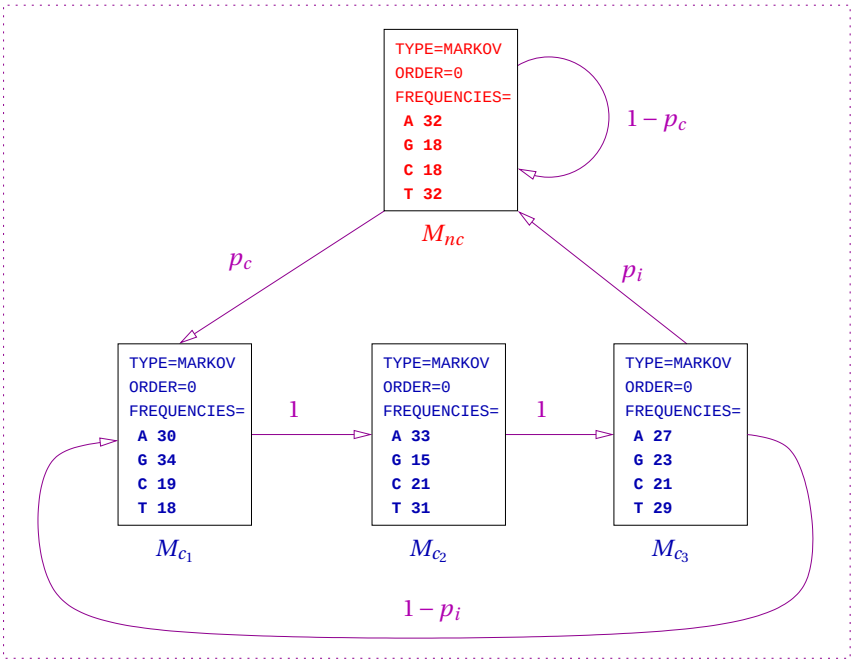


Fig. 5.3: Modèle de Markov caché pour l'alternance de régions codantes/intergéniques dans l'ADN de *Bacillus Subtilis*.

syntaxiques, c'est à dire ceux qui discriminent les séquences autorisées de celles interdites. Les expressions régulières sont un outil bien connu de la théorie des langages, qui permettent une description condensée d'un langage, c'est à dire d'un ensemble de séquences. Elles sont définies récursivement, de même que le langage qui leur est associé, de la façon suivante :

$$e = \begin{cases} e'^* & \Rightarrow & \mathscr{L}(e) = \mathscr{L}(e')^* \\ e' \mid e'' & \Rightarrow & \mathscr{L}(e) = \mathscr{L}(e') \cup \mathscr{L}(e'') \\ e' \,.\, e'' & \Rightarrow & \mathscr{L}(e) = \mathscr{L}(e') \times \mathscr{L}(e'') \\ (e') & \Rightarrow & \mathscr{L}(e) = \mathscr{L}(e') \\ \alpha & \Rightarrow & \mathscr{L}(e) = \{\alpha\} \\ \varepsilon & \Rightarrow & \mathscr{L}(e) = \{\varepsilon\} \end{cases}$$

Où e' et e'' sont des expressions régulières et α est un symbole du vocabulaire terminal Σ.

Des mots d'une taille n, donnée appartenant au langage dénoté par l'expression régulière peuvent être engendrés uniformément en temps $\mathscr{O}(n)$ (voir [47] ou [45]). De plus, nous avons transposé le système de pondération des symboles terminaux introduit en section 3.3 et définis en page 29, afin de contrôler la proportion des symboles.

Exemple :
L'expression régulière

$$e = (A.T.G).((A|T|G|C).(A|T|G|C).(A|T|G|C))^*.(T.A.G|T.G.A|T.A.A)$$

décrit un modèle simple pour une région codante, ou ORF, débutant par un codon START (ATG), prolongé par une séquence de codons non-contraints $((A|T|G|C).(A|T|G|C).(A|T|G|C))^*$ et terminant par un des trois codons STOP TAG, TGA ou TAA. On notera qu'une telle expression n'empêche pas l'apparition d'un codon STOP au sein des séquences engendrées.

Exemple :
Une expression régulière peut aussi être employée pour coder efficacement des mutations. Partons de ces trois séquences d'ARN ribosomal 5.8S, proches les unes des autres et alignées comme suit :

...	C	G	C	C	C	C	G	C	C	G	G	C	G	G	...
...	A	C	G	C	G	A	C	C	C	G	G	U	G	G	...
...	C	C	U	G	U	U	-	G	U	G	G	U	G	G	...

Une expression régulière peut alors être utilisée pour modéliser une famille de séquences contenant les trois séquences d'origine, de même que de nombreuses autres séquences proches des originaux.

$$e = \cdots (C|A).(C|G).(C|G|U).(C|G).(C|G|U).(C|A|U).(G|C|\varepsilon).(C|G).(C|U).G.G.(C|U).G.G\cdots$$

Pondérations

Comme dans le cas des grammaires pondérées, il est possible d'utiliser des poids associés aux symboles terminaux pour altérer la distribution des séquences. Ce système de pondération permet de contraindre la proportion de chaque symbole terminal dans les séquences engendrées. On trouvera dans [29] une résolution analytique du calcul des pondérations réalisant des proportions (ou fréquences) attendues.

Motifs PROSITE : Origine et fonction

PROSITE est une base de données consacrée aux différentes familles de protéines et aux domaines protéiques. Elle consiste en une collection de sites, de motifs et de profils aidant à identifier à quelle famille de protéines appartient une séquence donnée. Elle a été initiée en 1989 par Amos Bairoch, et fait partie du programme SWISS-PROT [8]. Elle contient des fichiers texte structurés par des balises et des retour-chariots. Parmi ces balises, certaines sont utilisées pour définir des *motifs*, qui établissent un lien entre une propriété de la séquence et une propriété fonctionnelle. On peut les interpréter comme des séquences *consensus*, dans la mesure où elles focalisent sur des similarités à l'intérieur des ensembles de séquences à partir desquels ils sont construits.
De plus amples informations peuvent être trouvées à l'adresse :

`http://www.expasy.org/prosite/prosuser.html`

Pris en temps que formalisme syntaxique, on peut dire des motifs PROSITE que leur expressivité est strictement incluse dans celle des expressions régulières.

Génération en taille non fixée

De plus, les ensembles de séquences dénotés par ces motifs sont toujours de cardinalités finies, ce qui ne les empêche pas d'être très souvent impossibles à manipuler explicitement. Le cardinal de l'ensemble de séquences engendrées par un motif est en effet potentiellement exponentiels sur la longueur du motif. Cependant, cette finitude nous permet d'envisager une approche récursive alternative, dans laquelle la taille des séquences à engendrer n'est plus fixée, et la séquence choisie uniformément sur l'ensemble des séquences dénotées. Cette approche a été implémentée dans `GenRGenS`, où l'on peut choisir ou non de tenir compte de la taille spécifiée pour les séquences.

Syntaxe et sémantique

Soit $\mathscr{P}$ l'ensemble des acides aminés connus, ramenés à leur codes IUPAC standards[3], de la figure 5.3.2. On peut alors définir récursivement les motifs PROSITE, ainsi que les ensembles de séquences associés, comme suit :

$$p = \begin{cases} p'\text{-}p'' & \Rightarrow \quad \mathscr{L}(p) = \mathscr{L}(p') \times \mathscr{L}(p'') \\ p'(n) & \Rightarrow \quad \mathscr{L}(p) = \mathscr{L}(p')^n \\ p'(n_{min}, n_{max}) & \Rightarrow \quad \mathscr{L}(p) = \mathscr{L}(p')^{n_{min}} \times (\{\varepsilon\} \cup \mathscr{L}(p'))^{n_{max}-n_{min}} \\ (p') & \Rightarrow \quad \mathscr{L}(p) = \mathscr{L}(p') \\ [l] & \Rightarrow \quad \mathscr{L}(p) = \{l_1, \ldots, l_{|l|}\} \\ \{l\} & \Rightarrow \quad \mathscr{L}(p) = \mathscr{P} \backslash \{l_1, \ldots, l_{|l|}\} \\ \mathtt{x} & \Rightarrow \quad \mathscr{L}(p) = \mathscr{P} \\ l & \Rightarrow \quad \mathscr{L}(p) = \{l\} \end{cases}$$

où p' et p'' sont des motifs PROSITE, n, n_{min} et n_{max} sont des entiers positifs et l est une séquence d'acides aminés.

5.3.3 Grammaires non contextuelles pondérées

Les grammaires non-contextuelles pondérées sont présentées formellement en section 3.2.2, définition 9 et sont étudiées tout au long de la section 4. Rapidement, on peut en dire qu'il s'agit de grammaires non-contextuelles $\mathscr{G}$, formalisme syntaxique qui discrimine par un processus génératif des séquences reconnues et interdites, auquel vient se greffer un système de pondération π induisant une distribution sur les séquences potentiellement engendrées. On trouvera dans la figure 5.5 un exemple de grammaire, la grammaire des mots de Dyck ou mots *bien parenthésés*, et dans la figure 5.6 la dérivation d'une séquence dans cette grammaire.

[3] Voir *http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html* pour plus de détails

Alanine	A
Aspartic acid ou asparagine	B
Cysteine	C
Aspartic acid	D
Glutamic acid	E
Phenylalanine	F
Glycine	G
Histidine	H
Isoleucine	I
Lysine	K
Leucine	L
Methionine	M
Asparagine	N
Proline	P
Glutamine	Q
Arginine	R
Serine	S
Threonine	T
Valine	V
Tryptophan	W
Inconnu	X
Tyrosine	Y
Glutamic acid ou glutamine	Z

Fig. 5.4: Codes standards pour les acides aminés, conformément à la proposition de l'IUPAC.

1	S	$\rightarrow$	$aSbS$
2	S	$\rightarrow$	ϵ

Fig. 5.5: Une grammaire non-contextuelle pour les mots bien parenthèsés.

$$\begin{aligned}\underline{S} &\underset{1}{\rightarrow} \mathbf{a\underline{S}bS} \underset{1}{\rightarrow} a\mathbf{a\underline{S}bS}bS \underset{2}{\rightarrow} aab\underline{S}bS \underset{1}{\rightarrow} aab\mathbf{a\underline{S}bS}bS \underset{2}{\rightarrow} aabab\underline{S}bS \\ &\underset{2}{\rightarrow} aababb\underline{S} \underset{1}{\rightarrow} aababb\mathbf{a\underline{S}bS} \underset{1}{\rightarrow} aababba\mathbf{a\underline{S}bS}bS \\ &\underset{2}{\rightarrow} aababbaab\underline{S}bS \underset{2}{\rightarrow} aababbaabb\underline{S} \underset{2}{\rightarrow} aababbaabb\end{aligned}$$

Fig. 5.6: Dérivation de la séquence $aababbaabb$ à partir de l'axiome S. Le symboles nouvellement créés au cours d'une dérivation sont en **gras**, et ceux qui seront réécrits à la prochaine étape soulignés.

1	S	$\rightarrow$	$aTbS$
2	S	$\rightarrow$	cS
3	S	$\rightarrow$	ϵ
4	S	$\rightarrow$	$aTbS$
5	S	$\rightarrow$	cS

Fig. 5.7: Une grammaire pour un sous-ensemble des mots de Motzkin en bijection avec les structures secondaires d'ARN.

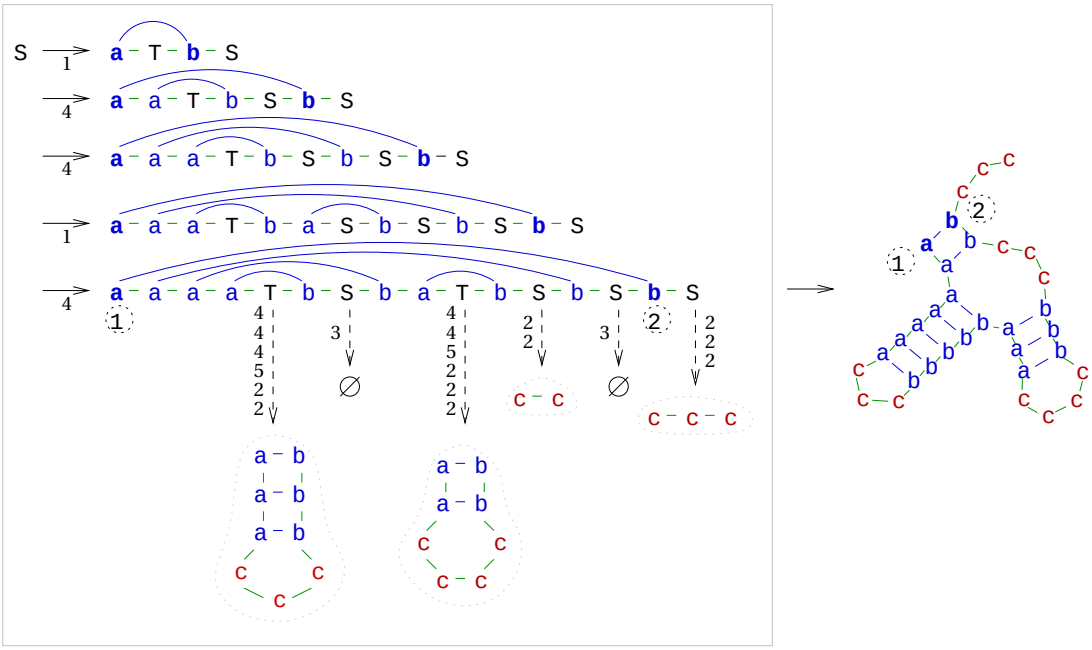


Fig. 5.8: Les bases **1** et **2**, bien qu'engendrées à un même instant, se retrouvent éloignées dans la séquence finale. On peut grâce à ce type de mécanisme tenir compte de la proximité géographique, constatée dans la structure secondaire équivalente visible dans la partie droite de la figure, de ces deux bases.

Utilisées comme des modèles de séquences structurées, les grammaires non-contextuelles permettent la modélisation d'*interactions longues distances*, c'est à dire de phénomènes impliquant conjointement des bases (ou symboles) situés à des distances non-bornables dans la séquence d'arrivée. La figure 5.8, qui simule une dérivation à partir de la grammaire des structures secondaires d'ARN présentée dans la figure 5.7, illustre ce mécanisme. On observe que des bases engendrées au cours d'une même étape peuvent être séparées par les réécritures successives. Cette proximité de génération autorise la prise en compte de leurs natures respectives dans leur contributions au *poids total* de la séquence. De plus, la possibilité d'adjoindre un modèle de Markov à toute grammaire non-contextuelle, résultat établi dans la section 4.3, prouve la capacité de ces modèles à prendre en compte des dépendances séquentielles classiques. Ce for-

malisme permet donc une prise en charge de toutes les dépendances induites par des liaisons (hydrogène ou phosphodiester) comprises dans la structure secondaire d'un ARN.

5.3.4 Modèles hiérarchiques

On l'a vu, les modèles de Markov cachés trouvent leur origine dans la nécessité de décomposer les séquences génomiques en régions présentant des propriétés différentes, et donc nécessitant des modélisations différentes. Les grammaires non-contextuelles se prêtent aussi admirablement à cet exercice, par l'emploi de non-terminaux différents pour les différentes régions. Cependant, il nous manquait un formalisme pour combiner *naïvement* ces différents modèles. C'est la raison d'être des modèles hiérarchiques, qui permettent une approche hiérarchique multi-modèles.

Le fonctionnement d'un modèle hiérarchique est le suivant :

- Une première séquence *maîtresse* ω_m de taille n, est engendrée sur un vocabulaire Σ_m selon un modèle M_m.
- Chaque occurrence dans ω_m d'un symbole $\alpha \in \Sigma_m$ faisant l'objet d'une réécriture $\alpha \rightarrow (M_\alpha, f_\alpha)$ est remplacée par une séquence de taille $f_\alpha(n)$ issue de M_α. On obtient ainsi la séquence ω_r qui est renvoyée par le logiciel.

Remarque 1 : Le traitement des occurrences de Σ_m n'est pas récursif, c'est à dire qu'un symbole de Σ_m obtenu lors de la réécriture d'un symbole α ne fera pas l'objet d'une réécriture.
Remarque 2 : La taille n fournie en argument au générateur n'a qu'une influence très indirecte sur la taille $|\omega_r|$ de la séquence engendrée.

Les fonctions de tailles f_α associées aux symboles de Σ_m sont définies récursivement à partir des opérations arithmétiques classiques (`+`,`-`,`/`,`*`,`^`), des fonctions utilitaires (`min`, `max`, `floor` ou `ceil`), de générateurs aléatoires de nombres (Lois uniformes sur $[n_{\min}, n_{\max}]$ et normales de paramètres μ et σ) ainsi que par un système de variables permettant des *effets de bord*, des dépendances entre les tailles des séquences issues des réécritures successives. Ces dépendances utilisent la pseudo-fonction `VAR`(v,e), qui associe une variable v aux évaluations successives d'une expression e. Cette variable peut alors être utilisée dans l'évaluation des autres expressions. Ce mécanisme permet d'imprimer des contraintes *transversales*, comme : *La somme des tailles de ces deux régions est un multiple de trois* ou bien *Les tailles de deux régions A et B successives sont, en moyenne, égales, mais peuvent varier d'un facteur* ε.

Afin d'illustrer ces différents aspects, on propose un modèle simple pour un site de décalage de phase -1 ribosomique basé sur une tige-boucle dans un ARN messager. Un site de décalage de phase provoque stochastiquement un comportement inhabituel du ribosome, qui *glisse* ou *patine* sur un motif structuré, ce qui modifie sa phase de lecture. La protéine engendrée n'est alors plus uniquement dépendante de la séquence de l'ARNm. On s'inspire ici d'un modèle de site observé dans le HIV-1 et basé sur la succession d'un heptamer, caractéristique de ce phénomène, d'une séquence *glissante*, appelée *espaceur* et d'une structure secondaire en *tige-boucle*.

L'**heptamer**, dont la séquence est contrainte par un consensus XXXYYYZ, sera modélisé par une expression régulière (Modèle M_h).
L'**espaceur** ayant une structure moins contrainte et comprise, sera modélisé par une chaîne de Markov (Modèle M_s).
La **tige-boucle**, nécessitant des interactions courtes et longues distances, sera économiquement modélisée par une grammaire non-contextuelle, éventuellement pondérée (Modèle M_t).
Les sites-candidats seront alternés avec un modèle Markovien triphasé général pour l'ARN messager(Modèle M_a).
L'alternance des modèles ci dessus sera gouvernée par un modèle maître très simple, basé sur

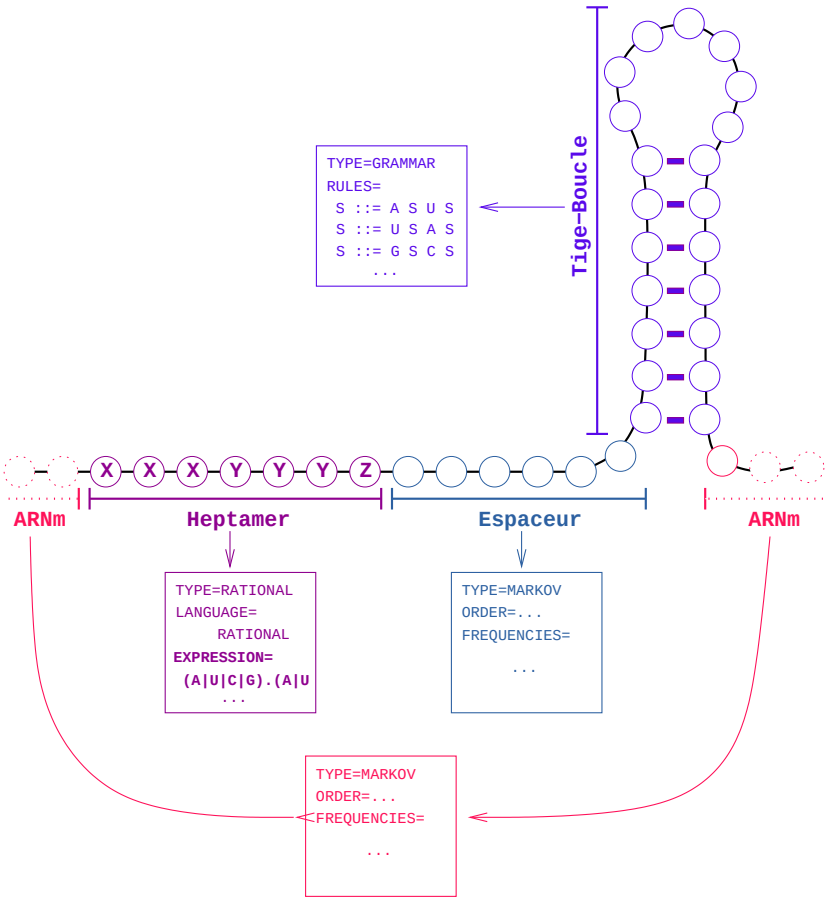


Fig. 5.9: Modèle de décalage de phase -1 basé sur une tige-boucle inspiré de VIH-1.

l'expression régulière :

$$e_m = (\texttt{heptamer.espaceur.tige-boucle.ARNm})^*$$

Les réécritures sont alors les suivantes :

```
   heptamer  →  (M_h,VAR(X,7))
   espaceur  →  (M_e,VAR(Y,UNIFORM(5,8)))
tige-boucle  →  (M_t,VAR(Z,UNIFORM(24,33)))
       ARNm  →  (M_a,3*FLOOR((NORMAL(100,302)+X+Y+Z)/3)-X-Y-Z)
```

L'utilisation du mot-clé `VAR` permet le stockage des évaluations des tailles dans des variables `X`, `Y` et `Z`. On peut ainsi garantir que les sites de frameshift apparaîtront tous dans la même phase, ce qu'on réalise dans la dernière formule, un peu cabalistique, pour la taille des réécritures du symbole `ARNm`. Concrètement, elle permet la génération d'un entier a compris entre 100 et 302 tel que la somme a+X+Y+Z soit multiple de 3.

5.3.5 Perspectives

GenRGenS est déjà intégré à certaines méthodes bioinformatiques [80; 50; 13; 2; 7; 93; 94]. Il lui manque encore la prise en charge de certaines classes de modèles pour une utilisation massive par la communauté.

Tout d'abord, on pourrait s'intéresser au **shuffling**, qui consiste à mélanger les lettres d'une séquences d'origine, et qui jouit d'une certaine popularité. De nombreuses variantes existent, allant jusqu'à soulever des problématiques algorithmiques complexes et intéressantes. Par exemple, le fait de préserver les nombres d'occurrences des différents k-lets peut être assimilé à la génération aléatoire uniforme d'un chemin eulérien dans un graphe[4]. On pourrait aussi tenter l'intégration du logiciel SMACK de R. Rivière et al., qui engendre aussi dans ce modèle en intégrant des contraintes de type *motif exclu* [81].

Ensuite, les **grammaires stochastiques**, transposée de l'étude du langage à la bioinformatique par Eddy et Durbin [38] méritent notre attention, car elles restent au coeur de nombreuses méthodes bioinformatiques orientées vers la structure des macromolécules. Leur génération pose cependant un problème quant à sa finitude. Pour certaines classes de grammaires stochastiques, le processus de génération diverge et il n'est pas possible de le stopper arbitrairement comme dans le cas des chaînes de Markov, sous peine d'obtenir une séquence immature, présentant encore de nombreuses occurrences de non-terminaux. Il pourrait être intéressant d'étudier la convergence des grammaires stochastiques apprises automatiquement, à partir de structures d'ARN [38] ou plus généralement.

[4] Ce formalisme avait déjà fait l'objet d'une implémentation publiée [25], mais elle n'est à notre connaissance plus accessible...

Partie II

ETUDES DE L'ARN ET SES STRUCTURES

Plan et contributions

Ce deuxième chapitre traite principalement de l'application des méthodes et techniques introduites précédemment à la structure de l'ARN. Nous procédons tout d'abord à un bref rappel des notions fondamentales de la biologie moléculaire, en mettant l'accent sur la nécessaire prise en compte de la structure de l'ARN par les méthodes bioinformatiques visant à élucider sa fonction.

Nous nous intéressons alors à la génération aléatoire de structures secondaires d'ARN *réalistes*. Après avoir rappelé et illustré les insuffisances du modèle uniforme classique pour ces structures (Section 7.1.1, page 98), nous proposons des grammaires de complexités et d'expressivités croissantes pour la structure secondaire d'ARN (Section 7.2, page 102). Nous nous intéressons alors à l'adjonction de pondérations aux grammaires proposées, et en particulier aux valeurs des pondérations pour lesquelles des propriétés observées dans des familles d'ARN sont réalisées par les structures engendrées. Dans les cas analytiquement *simples* des tiges-boucles (ou micro-ARN) (Section 7.3.1, page 105) et des structures secondaires (Section 7.3.2, page 107) pour lesquelles seule sont considérées les proportions de bases appariées et non-appariées, des résultats analytiques exacts sont obtenus. Dans le cas de paramètres plus complets et complexes (nombres et tailles moyennes des sous-structures), le prototype `GrgFreqs` introduit au chapitre premier permet le calcul de pondérations compatibles avec des valeurs pour les paramètres observées dans des familles connues d'ARN (Section 7.4, page 108).

Enfin, nous nous intéressons à l'obtention des données de structures secondaires d'ARN. Les méthodes fiables pour la détermination de la structures des ARN étant principalement expérimentales, il est nécessaire de pouvoir tenir compte de ces données. Cependant, celles-ci font apparaître des interactions, principalement non-planaires, incompatibles avec les outils algorithmiques disponibles. Nous nous interrogeons alors sur l'extraction d'une quantité maximale d'information dans une structure générale d'ARN issue de méthodes expérimentales. Nous proposons principalement deux piste : Extraire la sous-structure planaire comprenant le plus grand nombre d'appariements, ou extraire la sous-structure d'énergie libre minimale, dans un modèle simple d'énergie libre (Nussinov-Jacobson ou Turner). Dans le cas du nombre maximal d'appariements, un algorithme est proposé(Section 8.3, page 114), qui se transpose naturellement au modèle de Nussinov, et le problème est rapproché du classique MAX-INDEPENDENT-SET dans les graphes circulaires.

Contributions :

- Calcul des nombres et propriétés des sous-structures dans les différents ARNs ribosomaux des différents règnes
  Figure 7.4 page Figure 101

- Grammaire non-contextuelle simple pour la structure secondaire d'ARN permettant l'exigence de tailles minimales pour les boucles et helices, ainsi que la contrainte ultérieure des nombres et tailles moyennes de ces deux sous-structures
  Section 7.2.2, page 103

- Grammaire complète pour la structure secondaire d'ARN, permettant le marquage des différents types de sous-structures
  Section 7.2.3, page 104

- Résolution analytique de la relation pondérations/proportions de bases appariées/non-appariées dans le cas des tiges-boucles et des structures secondaires
  Section 7.3.1, page 105 et Section 7.3.2, page 107

- Conception et implémentation d'un algorithme dédié à l'extraction d'un sous-ensemble planaire d'appariements dans une structure générale
  Section 8.3, page 114

6. NOTIONS DE BIOLOGIE MOLÉCULAIRES

6.1 Les macromolécules

Les macromolécules, terme regroupant l'ADN, les ARN et les protéines, sont au coeur des hypothèses actuellement proposées pour expliquer le vivant.

6.1.1 L'ADN

L'ADN est le support de l'information génétique. Et, bien que son étude hors-contexte ne suffise pas à expliquer son rôle prédominant dans les mécanismes cellulaires, celle ci constitue un premier champs d'investigation fructueux. On présentera donc dans un premier temps l'ADN *nu* ou *au repos* en temps que macromolécule régie par des règles chimique, sans se soucier de ses interactions avec son milieu. On évoquera ensuite quelques dynamiques dans laquelle il est impliqué, et particulièrement la transcription, à l'origine des ARN.

Dans une définition biochimique, l'ADN est un polymère unidimensionnel, c'est à dire une séquence de monomères, les *nucléotides*.

Les désoxyribonucléotides

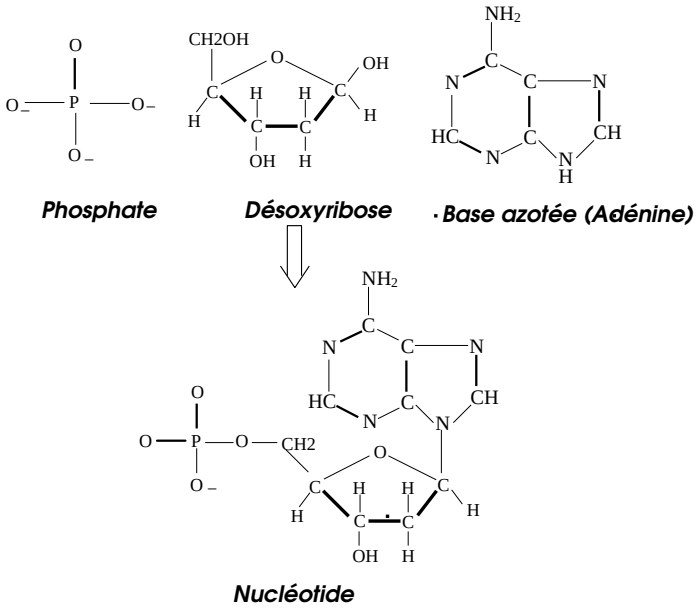


Fig. 6.1: Constitution d'un désoxyribonucléotide (Adénosine-phosphate)

Un nucléotide est une combinaison covalente d'un acide phosphorique, d'un sucre et d'une base azotée parmi l'Adénine(A), la Guanine(G), la Cytosine(C) et la Thymine(T). Dans le cas de l'ADN, le sucre est un désoxyribose, on parle donc de désoxyribonucléotides . On sépare ces bases en deux catégories, les bases puriques (adénine et guanine), qui contiennent 2 cycles et

les base pyrimidiques(cytosine, thymine[1]), qui ne contiennent qu'un cycle.

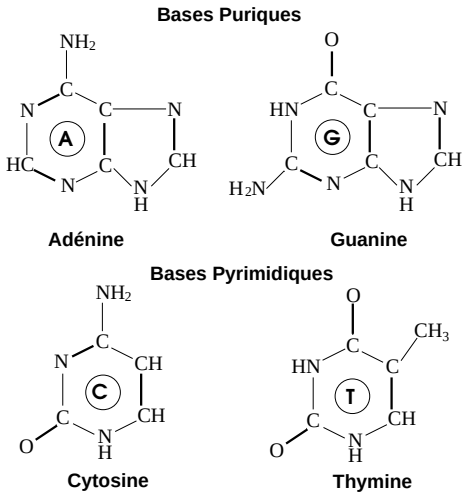


Fig. 6.2: Les bases azotées.

Appariements des bases azotées dans l'ADN

La structure chimique des bases donne lieu à des assemblages de bases dites complémentaires, formant des cycles pyrimidiques/puriques. Les bases complémentaires, classiquement, dans l'ADN sont l'Adénine-Thymine (A/T) et la Cytosine-Guanine (C/G). Les appariements qui en découlent sont appelés appariements Watson Crick. L'union A/T est consacrée par deux liaisons hydrogène, tandis que l'appariement C/G met en jeu trois liaisons hydrogène.

La double hélice

Les nucléotides s'empilent dans la séquence au moyen de liaisons phosphodiester, qui participent à l'assemblage, grâce à un phosphate, des désoxyriboses des nucléotides. Les nucléotides s'assemblent ainsi en de longues séquences orientée 5'/3' par héritage des polarités des désoxyriboses, polarités illustrées par la figure 6.4.
La structure tridimensionnelle de l'ADN est une double hélice[101] particulièrement stable, régulière et relativement insensible à la composition en bases. Il est donc admis que l'information génétique contenue dans l'ADN est plus à chercher dans la séquence de ses base que dans sa structure tridimensionnelle ou tertiaire[2]. On étudie donc d'abord la structure primaire de l'ADN, qui

[1] L'Uracile, substitut dans l'ARN de la thymine, est aussi une base pyrimidique

[2] Cependant, l'étude de l'enroulement de la double hélice autour d'un histone, appelé nucléosome, à permis de mettre en évidence des interactions entre paires de bases proches géographiquement, ce qui plaide pour une importance de la structure tertiaire de l'ADN. Notamment, des périodicités de 200bp sont observées lors de l'observation dans les séquences, ce qui correspond à la circonférence des nucléosomes.

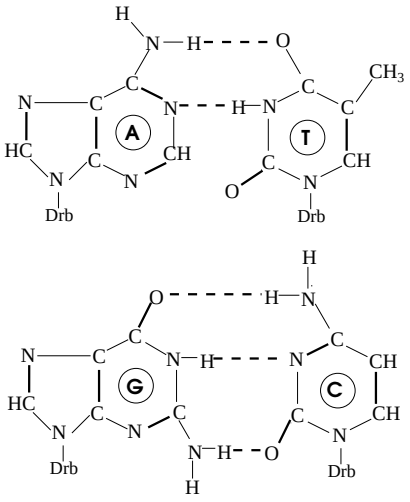


Fig. 6.3: Les appariements Watson Crick.

correspond à la séquence des bases sur un brin lu dans l'ordre 5' vers 3'. Les bases du brin complémentaires sont entièrement déterminées par complémentation, on peut donc les ignorer lors d'un traitement algorithmique. En effet, dans l'ADN, les seuls appariements possibles sont A/T et C/G pour des problèmes de stabilité et d'encombrement stérique.

6.1.2 L'Acide Ribonucléique (ARN)

L'ADN est circonscrit au noyau de la cellule chez les eucaryotes. Or l'expression phénotypique du matériel génétique passe essentiellement par l'action de protéines, qui sont localisées hors du noyau, dans le cytoplasme.
L'ARN est donc entre autre l'intermédiaire par lequel l'ADN s'exprime en dehors du noyau. Il est synthétisé à partir d'un segment d'ADN.
Tout le contenu informatif de l'ADN est transcrit en un ARN qui sera soit un acteur (ARN ribosomaux ou ARN de transfert), soit une information *brute* nécessitant étape supplémentaire de traduction pour s'exprimer (ARN messagers).

Définition bio-moléculaire de l'ARN

La molécule d'ARN est, comme l'ADN, un polymère linéaire dont les monomères sont cette fois ci des ribonucléotides, par substitution du sucre β-D-ribose au désoxyribose. Comme le montre la figure 6.5, un ribonucléotide est la combinaison covalente d'un phosphate, d'un ribose et d'une base azotée. Les bases azotées disponibles pour l'ARN sont l'Uracile, l'Adénine, la Guanine et la Cytosine. L'orientation de la molécule d'ARN est la même que celle de l'ADN, du 5' au 3'.

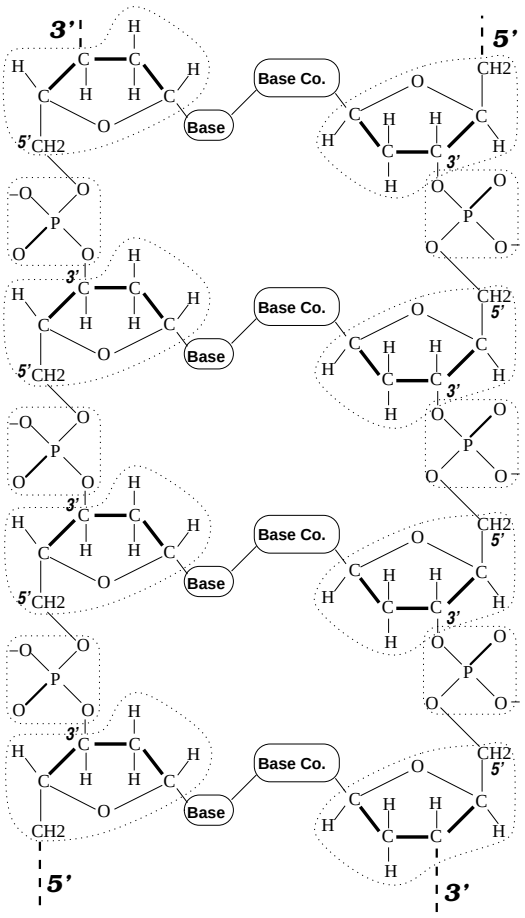


Fig. 6.4: Assemblages de nucléotides formant la double hélice.

Repliement de l'ARN

Les structures secondaires et tertiaires de l'ARN sont bien plus variées et porteuses d'informations fonctionnelles que celles de l'ADN. En effet, la structure simple brin de l'ARN permet un repliement de la molécule sur elle même, au moyen d'appariements Watson-Crick(voir 6.3 en substituant l'uracile à la thymine), mais aussi d'appariements Wobble ou Hoogsteen (voir 6.7). On trouvera en [56] un travail de classification des appariements.
Le système de liaisons ainsi défini, associé aux dispositions des ribonucléotides dans les différents types d'appariements, définit la conformation spatiale de l'ARN.
Dans de nombreux cas, on met en évidence une relation entre cette structure de l'ARN et son intervention dans les mécanismes biologiques. Cette structure est donc porteuse d'information sur la fonction de l'ARN, au même titre que la séquence des bases.
De plus, il n'existe pas d'inclusion stricte entre les contenus informatifs de la séquence et ceux de la structure. En effet, on ne peut pas associer à une conformation spatiale donnée une unique

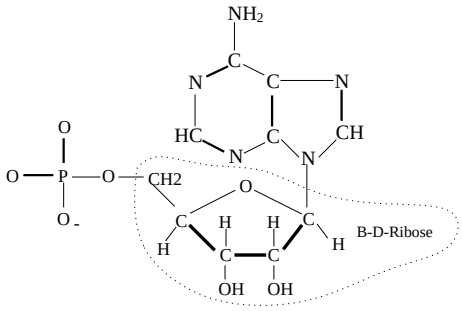


Fig. 6.5: Un ribonucléotide

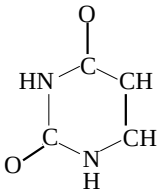


Fig. 6.6: L'uracile, substitut de la thymine dans les ARNs

structure primaire, de même qu'il semble impossible d'établir le repliement tridimensionnel de l'ARN avec certitude à partir de la structure primaire. L'étude conjointe de la séquence de bases et de la structure semble donc nécessaire à la compréhension du rôle d'un ARN. La complexité, prise ici au sens algorithmique, de l'étude de la structure tertiaire étant rédhibitoire[3], on ne considère souvent que la structure secondaire d'un ARN quand on veut prendre en compte sa topologie.

6.2 Le dogme central de la biologie moléculaire

On présente ici le *dogme central* de la biologie moléculaire, simplification pour certains excessives des mécanismes régissant le vivant. Il fait actuellement l'objet d'une remise en question assez radicale, notamment de la part des tenants du *monde à ARN*. On axera cependant l'introduction ci-dessous sur ce dogme fondamental, ceci autant pour son rôle historique que pour sa simplicité.

Selon ce dogme, l'ADN serait le support de l'information génétique, l'ARN un intermédiaire intervenant parfois dans des mécanismes cellulaires et la protéine l'acteur du monde extranucléaire.

[3] L'apparition d'interaction tertiaire simple, comme les pseudo-noeuds fait sortir les structures étudiées de l'univers *hors contexte*, ce qui pénalise un traitement récursif. Par exemple, le repliement de l'ARN avec des pseudo noeuds généralisés est déjà un problème NP-Complet[58] dans le modèle d'énergie libre simple de Turner[106].

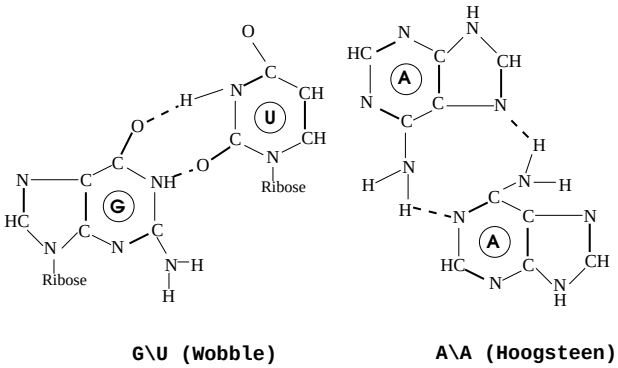


Fig. 6.7: Quelques appariements non Watson Crick observables chez certains ARNs

6.2.1 La réplication

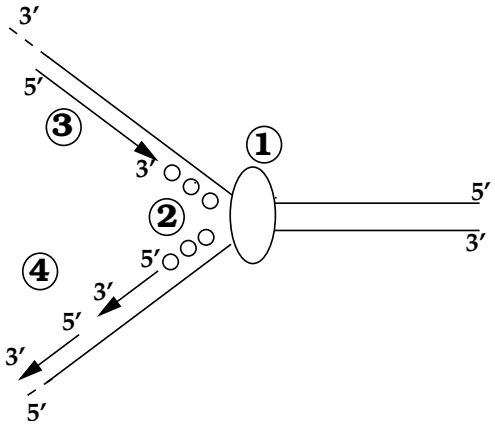


Fig. 6.8: Réplication de l'ADN

Lors de la mitose, les deux brins de l'ADN sont séparés par une enzyme, l'hélicase (voire figure 6.8, 1). Son travail est complété par des protéines de liaisons (2), qui empèchent les deux brins de se recoller derrière l'hélicase. Les complémentaires, aussi appelés molécules filles, des deux brins ne peuvent, pour des raisons chimiques, être générés que dans le sens 5'→3'. Le brin père 3' est donc complété séquentiellement par la DNA-polymérase(3). Le brin 5' nécessite quant à lui la création, toujours par la DNA-polymérase, de petites séquences, les fragments d'Okazaki, qui seront ensuite recollés par une ligase(4).
La replication est, toujours chez les procaryotes et très souvent chez les eucaryotes, bidirectionnelle et initiée à partir de sites d'amorçage, des séquences de bases spécifiques. L'intervention d'une enzyme, la télomèrase, est ensuite nécessaire pour reconstituer les extrémités, appelées télomères, altérées par le mécanisme de réplication. Les télomères sont donc reconstitués de

façon quasi systématique par des copies d'un fragment d'ARN puis se replient en boucle pour éviter toute dégradation de l'ADN.

6.2.2 La transcription

On a jusqu'ici évoqué le support et la copie de l'information génétique, il convient maintenant d'en expliquer l'expression. L'ADN s'exprime par l'intermédiaire des ARN, auxquels nous consacreront le prochain chapitre, molécules qui sont créées au cours de la phase de transcription. Seules certaines portions de l'ADN sont transcrites, ces séquences sont appelés gènes.
Lors de la phase de transcription, un complexe protéique,l'ARN polymérase, se fixe sur l'ADN. On appelle promoteur le site de fixation de l'ARN polymérase. Il est de taille variable, et peut associer plusieurs séquences cibles. La plus fréquente est la séquence TATAAT, qui est propice à l'écartement des deux brins.La fixation de l'ARN polymèrase sur l'ADN peut être activée ou inhibée par la présence de protéines, appelées facteurs régulateurs de transcription.
Une fois l'ARN polymérase fixée, elle réalise une copie complémentaire d'une portion du brin sens de l"ADN dans le sens 5'→3', en complémentant le brin anti-sens dans le sens 3'→5' (voire Figure 6.10). Elle substitue au passage l'uracile à la thymine. La transcription est stoppée quand l'ARN polymérase rencontre un site particulier, appelé terminateur, ou sur intervention de facteurs protéiques chez certains procaryotes[4]. Le polymère obtenu est appelé transcrit primaire de l'ARN ou préARN. Ce transcrit présente une segmentation en 3 régions : 5'UTR, ORF, 3'UTR, l'ORF pouvant être interrompue par des introns
Chez les eucaryotes, il existe quelques aménagements au cadre général de la transcription :

- Chaque classe d'ARN est engendrée par une ARN polymérase spécifique.

- Avant d'être fonctionnel hors du noyau, le transcrit primaire doit faire l'objet d'une maturation. La maturation consiste, classiquement, en l'adjonction d'une *coiffe*[5] en 5', d'une queue polyadénilée en 3'[6], ainsi qu'en un épissage du transcrit primaire. Celui est en effet constitué d'une alternance d'exons, matériel génétique codant, et d'introns non codants. L'épissage est le mécanisme par lequel les introns sont éliminés du transcrit primaire comme le montre la figure 6.9 et les introns sont concaténés.

6.2.3 La traduction

La traduction est le mécanisme par lequel une protéine est engendrée à partir des informations contenues dans un ARN messager. Elle met en jeu une macro molécule complexe, le ribosome. Dans la suite, on appellera codon un triplet de nucléotides.

[4] Facteur rho chez E. Coli.
[5] Une coiffe est la somme d'un groupement triphosphate et d'une base purique (A ou G).
[6] Sauf chez les ARN codant pour des histones, protéines à la base du nucléosome.

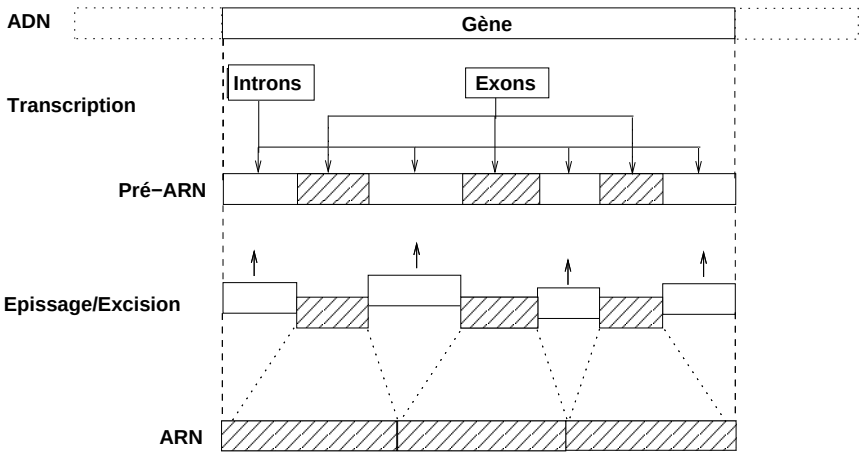


Fig. 6.9: L'épissage du transcrit primaire chez les eucaryotes

Le ribosome

Le ribosome est constitué de matériel mixte ARNr/protéines. On le décompose en deux partie, appelées petite et grande sous unités. Chacune des sous unités est composée d'ARNm(s) et de protéines. Hors de la phase de traduction, les sous unités des ribosomes sont séparés par la présence de protéines.

Initiation

Grossièrement, on peut dire que la petite sous unité du ribosome se fixe sur l'ARNm, ce qui permet à la plus grande sous unité de se coller à elle en entourant l'ARNm.
Le début de la traduction d'un ARN messager se fait à partir d'un codon spécifique, ou codon START(AUG,GUG ou UUG).
Chez les eucaryotes, le ribosome se fixe en amont du codon START. Il glisse alors dans le sens 5'→3' en scannant l'ARNm à la recherche d'un codon START.
Chez les procaryotes, l'ARN 16S se fixe sur l'ARNm en amont de moins de 10 bases du codon START. Cette zone contient un motif dit de Shine-Dalgarno qui correspond au complémentaire du 16S.
Une fois le codon START trouvé, un ARNt chargé spécifique de l'initiation, l'ARNti, se fixe sur le codon START et initie la traduction.

Elongation

Pendant la phase d'élongation, les acides aminés sont concaténés pour former des protéines. L'élongation est une séquence de formations de liaison peptidiques et de translocations.
Tout d'abord, le ribosome permet la fixation d'un ARNt chargé dans le site A. Une liaison peptidique se forme alors entre les acides aminés fixés à l'ARNt présents sur le site P et celui fixé sur

l'ARNt de A.
Intervient alors une étape de translocation, au cours de laquelle le complexe formé par l'ARNt de A et sa chaîne peptidique sont décalés sur le site P par une translation du ribosome d'exactement un triplet. L'ARNt initialement présent en A hérite alors de la chaîne peptidique présente en P. L'ARNt déchargé est alors éjecté. Le site A, libre et exposé à un nouveau codon, provoque l'arrivée d'un nouvel ARNt porteur de son anti codon et de son acide aminé correspondant. On se retrouve alors dans les conditions initiales de la translocation, qui se reproduit donc jusqu'à la terminaison.

Terminaison

La terminaison se produit quand le ribosome rencontre un codon STOP (UAG, UAA et UGA), qui ne correspondent à aucun anticodon d'ARNt. Le site A, vide, est disponible pour un facteur protéique RF (Release Factor) qui permet une dernière translocation. Cette translocation provoque une coupure entre l'ARNt présent en P et la chaîne polypeptidique dont il est le porteur. Les sous unités du ribosomes se séparent alors.

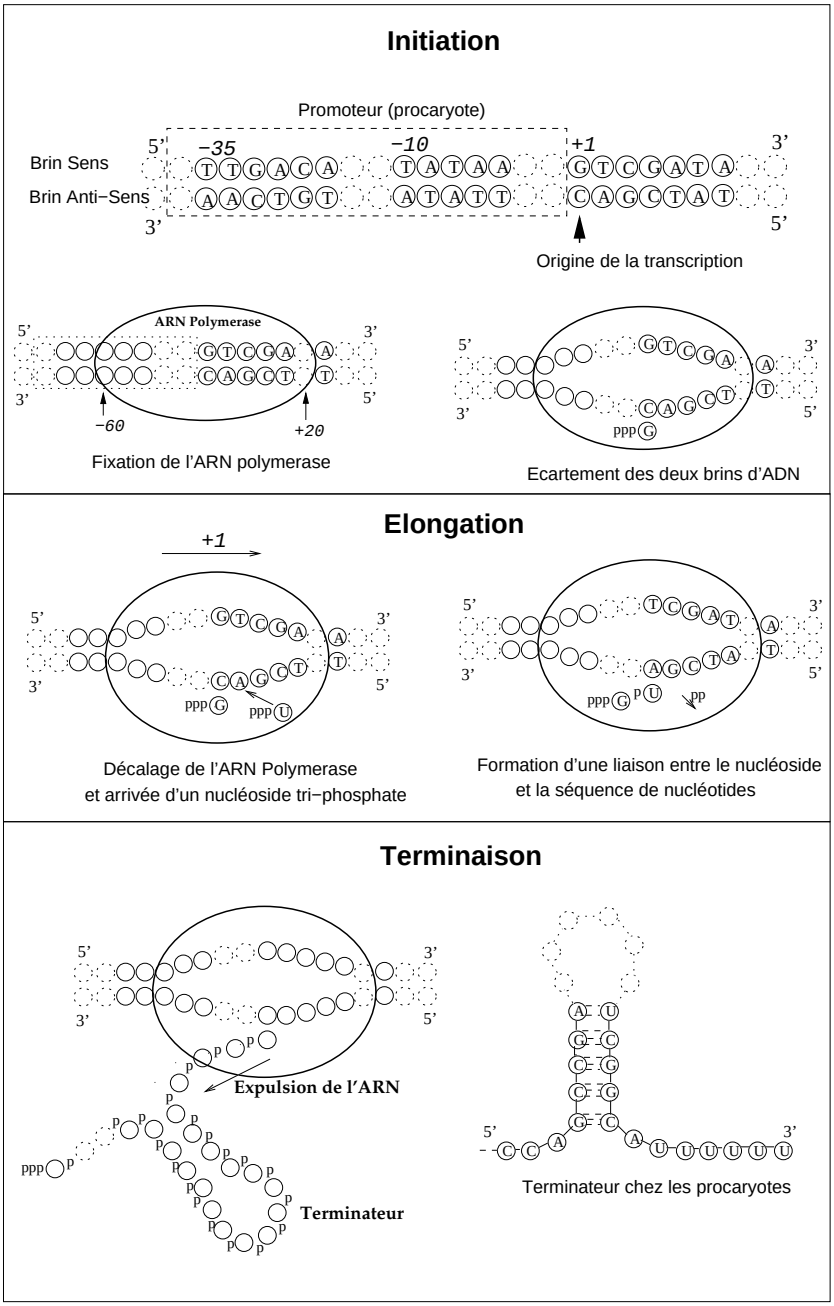


Fig. 6.10: Les étapes de la transcription.

Famille	*Sous-Unité*	**ARNr**	Nombre de protéines
Procaryotes	*Grande Sous-Unité*	**5S+23S**	34
	Petite Sous-Unité	**16S**	21
Eucaryotes	*Grande Sous-Unité*	**5S+5,8S+28S**	49
	Petite Sous-Unité	**18S**	33

Fig. 6.11: Les compositions des ribosomes

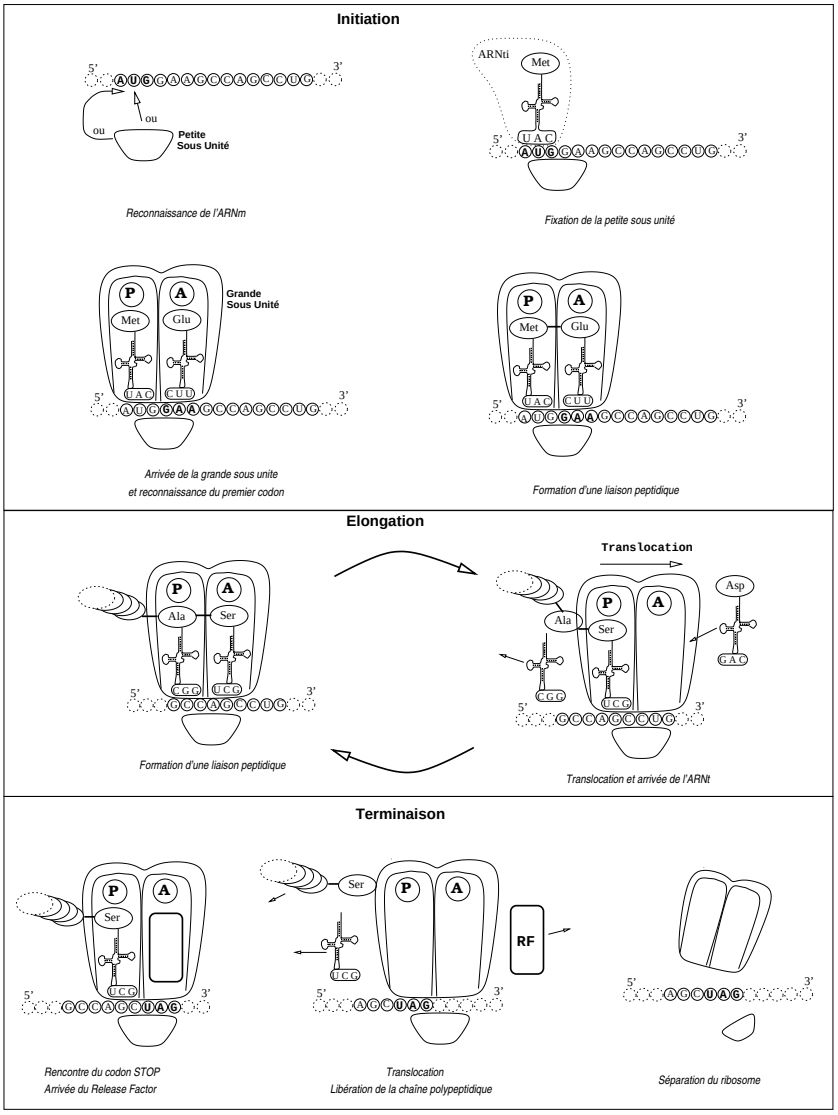


Fig. 6.12: Les phases de la traduction

6.3 Structures des ARN

6.3.1 Représentations de la structure d'une molécule

On caractérise l'organisation dans l'espace des macromolécules biologiques (ADN, ARN ou protéines) par leurs structures primaire, secondaire et tertiaire.

- **Structure primaire :**
  La structure primaire est la séquence des bases lues dans le sens 5'→3'.

Fig. 6.13: Structure primaire d'ARNr 16S de Pyrococcus Furiosus

- **Structure secondaire :**
  Il n'existe pas de consensus total sur la définition des structures secondaires.

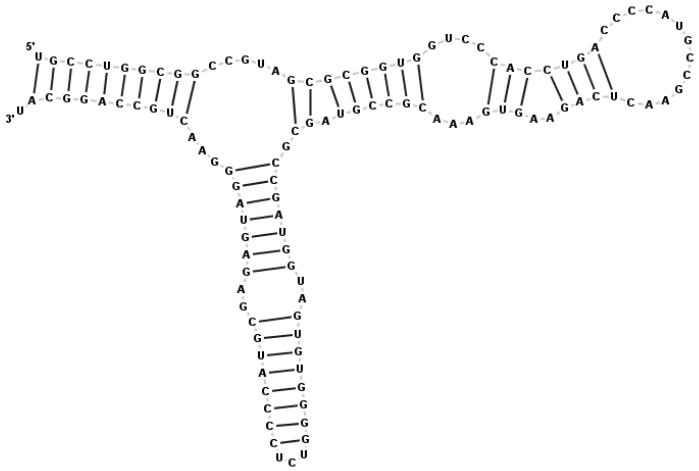


Fig. 6.14: Structure secondaire d'un ARNr 5s chez E. Coli

On peut cependant caractériser la structure secondaire d'une macromolécule par un ensemble de liaisons entre ses bases. De plus, afin de respecter une correspondance entre type de structure et dimension de l'espace requis pour les dessiner[7], on considère parfois

[7] primaire ⇔ linéaire (1D)
secondaire ⇔ planaire (2D)
tertiaire ⇔ tridimensionnelle (3D)

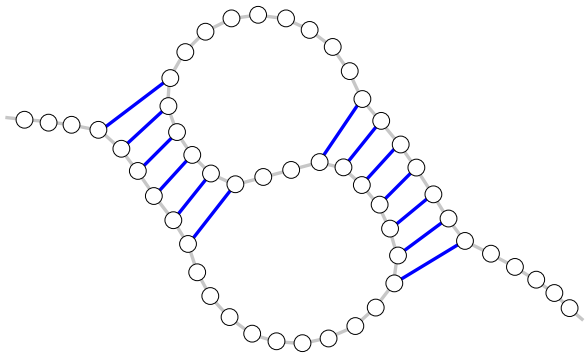


Fig. 6.15: Structure d'un pseudo noeud

qu'une structure secondaire se doit d'être planaire. Au delà de ce plus petit dénominateur commun, il reste à définir les types de liaisons autorisée, ainsi que la tolérance ou non de pseudo-noeuds (voir Figure 6.15). Il circule dans la communauté bioinformaticienne autant de définitions des structures secondaires que de variations[8] sur ces deux thèmes. On reviendra sur ce point dans la suite de ce document.

- **Structure tertiaire :**
  La structure tertiaire, est la localisation des constituants chimiques de la molécule dans

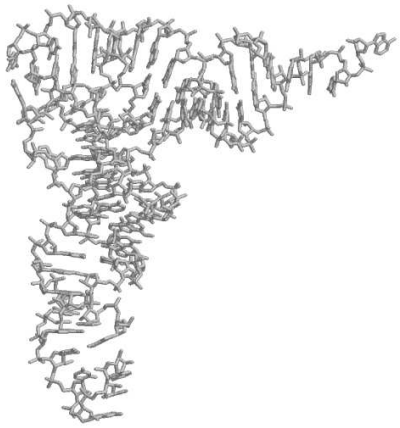


Fig. 6.16: Structure tertiaire d'un ARNt

l'espace. Là encore, certains se cantonne à la topologie de la molécule quand d'autres lui préfèrent les positions relatives des bases dans l'espace tridimensionnel.

[8] Des pseudo-noeuds *imbriqués* ne sont plus planaires. Certaines définitions admettent donc la présence de pseudonoeuds *raisonnables*, c'est à dire planaire.

- **Structure quaternaire:**
  On évoque parfois la structure quaternaire d'une molécule en indiquant son position-

Fig. 6.17: Structure quaternaire des histones

nement relativement à des molécules avec lesquelles elle est liée physiquement pour permettre une fonction. C'est le cas de la structure tridimensionnelle du nucléosome (voir Figure 6.17), décrite comme la structure quaternaire des histones.[9].

6.3.2 Structure secondaire de l'ARN : Discussion

Il n'existe pas de consensus total sur la définition des structures secondaires. On peut dégager les points suivants, parfois contradictoires, et classés *plus ou moins* par ordre de consensus décroissant :

1. La structure secondaire ne comprend pas les coordonnées tridimensionnelles des bases.
2. Elle considère un sous-ensemble des appariements.
3. Elle se restreint à des liaisons *canoniques*, définis comme telles du fait de leur forte énergie.
4. Elle exclut les liaisons triples.
5. Elle se limite à la donnée de liaisons, c'est à dire de données strictement topologiques.

[9] La structure tridimensionnelle du nucléosome est parfois décrite comme la structure quaternaire des histones.

6. Elle contient une forte majorité de l'énergie totale de la structure.

7. Elle est *planaire*, c'est à dire dessinable sur une surface plane sans croisement.

8. Elle n'est pas *unique* à travers le temps.

9. Elle *existe*.

Si les points 1 et 2 sont généralement admis dans la communauté s'intéressant à la structure de l'ARN, on peut facilement montrer que la condition de planarité en 7 peut théoriquement empêcher l'existence d'une structure secondaire contenant plus de quelques appariements. De plus, l'affirmation 7 est ambiguë : Parle-t-on du graphe de l'ARN, ou bien d'une *carte* donnée ? Un tel objet est construit à partir du graphe mais on opère au cours de cette construction des choix compromettant parfois la possibilité d'extraire un grand sous ensemble planaire d'appariements, comme l'illustre la figure 6.18.

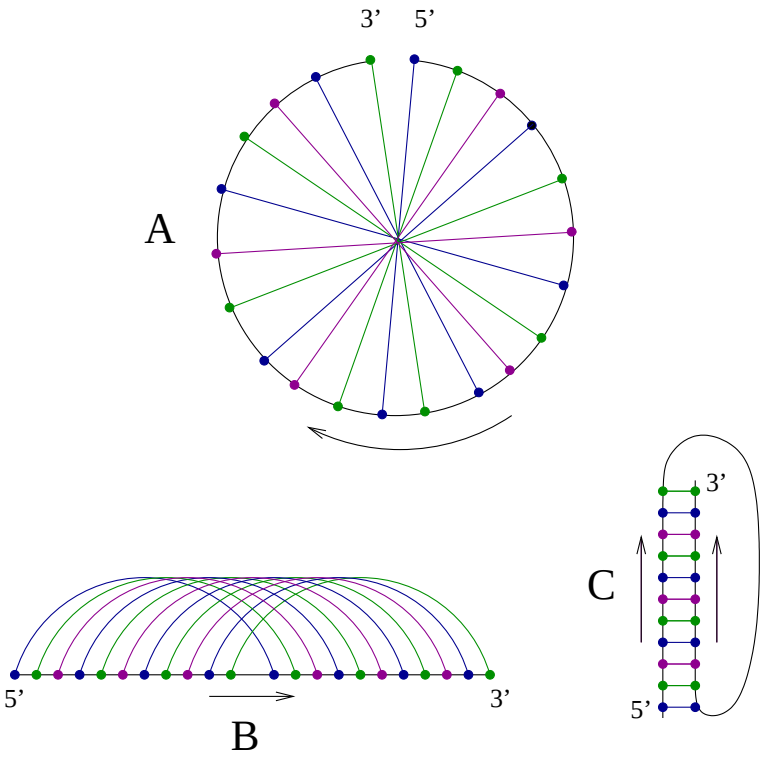


Fig. 6.18: Deux cartes **B** et **C** associées à un même graphe d'ARN **A**, et admettant des sous-ensembles planaires d'arcs de taille respectivement 1 et n.

En outre, l'adjectif *planaire* est souvent utilisé en lieu et place de *planaire extérieur*, terme désignant dans le cas des ARN un graphe pouvant être dessiné sans croisement en plaçant les arcs sur le demi-plan supérieur. Cette notion de *type de planarité* est cruciale pour des raisons algorithmiques. En effet, les graphes d'ARN planaires extérieurs admettent une décomposition

non-ambiguë extrêmement simple [100] alors que les structures bisecondaires, ou dessinables sur deux demi-plans, ne sont déjà décrits par aucun langage non-contextuel, ce qui complique la découverte de décomposition. On trouvera aussi dans [58] une preuve de NP-complétude du repliement de l'ARN dans le modèle de Turner quand les pseudonoeuds sont inclus.

Le dernier point est un rappel en forme de boutade du fait que la structure secondaire n'est, quelle qu'en soit la définition formelle, que la projection d'un instantané. Elle ne devrait donc, outre la remarque 8, ne pas être trop définitivement associée à un ARN donné.

6.3.3 Origine des données structurales

Cristallisation aux rayons X

Au cours de l'observation des macromolécules, les échelles mises en jeu rendent impossible l'observation optique classique. On peut alors avoir recours aux rayons X qui, de part leur longueurs d'ondes plus faibles, produisent un phénomène de diffraction au contact de la molécule. Les faibles longueurs d'ondes des rayonnements observés rendent difficile la focalisation, c'est pourquoi on travaille des structures cristallisées. Les cristaux sont en effet plus stables et offrent une organisation régulière, ce qui permet l'apparition de phénomènes d'interférences résultant de la diffraction de photons sur des facettes ayant même orientations et positions relatives.

Résonance magnétique nucléaire

De part leur spin non nul, les atomes d'hydrogène et de carbone peuvent être assimilés à de petits électro-aimants. On peut alors les exciter, les faire entrer en résonance, en les soumettant à un champs magnétique. Quand l'action de celui ci est interrompue, on peut mesurer par le biais d'une transformée de Fourier la distance à l'observateur des différents atomes et, en multipliant les mesures, retrouver leur positions initiales.

Génomique comparative

On part du principe que la structure de certains ARN est essentielle à sa fonction, et donc indirectement à l'adéquation de l'individu avec son milieu. On peut donc s'attendre à ce que les mécanismes de la sélection naturelle privilégient un faible taux de mutation dans les régions contribuant à la structure secondaire. On peut aussi s'attendre à ce que la mutation d'une base impliquée dans un appariement ne soit viable que dans le cas où la deuxième base de l'appariement mute aussi, rendant possible de nouveau l'appariement initial. Autrement dit, dans les hélices, les *mutations sont corrélées* au sein d'une paire de base. A partir d'un ensemble de séquences homologues suffisamment grand, on peut donc reconstruire la structure à partir des corrélations observées dans les probabilités de mutations des bases.

6.3.4 Fonctions et structures

On distingue les différents types d'ARN selon les rôles qu'ils jouent au sein de la cellule.

Les ARN messagers (ARNm)

Ils représentent moins de 5% des ARN cellulaires.
Rôle : Ils amènent l'information extraite de l'ADN au ribosome, qui va exprimer cette information par la synthèse protéique.
Structure : Les structures secondaires des ARN messagers semblent assez variées, aucun modèle fortement contraignant n'a jusqu'ici été formulé. Cependant, cette structure est essentielle car centrale au fonctionnement de la technique d'ARN interférence. De plus, la formation de motifs structurels spécifiques, comme des pseudo-noeuds, provoque l'apparition de phénomènes. Jusqu'à présent, on considérait que la structure des ARNm était négligeable. En effet, au cours de la traduction, l'ARN est étiré en une structure linéaire, ce qui semblait plaider pour une nullité du rôle joué par sa structure. Cependant, des études récentes sur le phénomène d'interférence d'ARN confère à l'ARNm un rôle qui dépasse celui d'intermédiaire et qui ferait intervenir sa structure. Cet intérêt pour la structure de l'ARNm étant assez récent, il n'existe pour l'instant que relativement peu de structures secondaires connues[10].

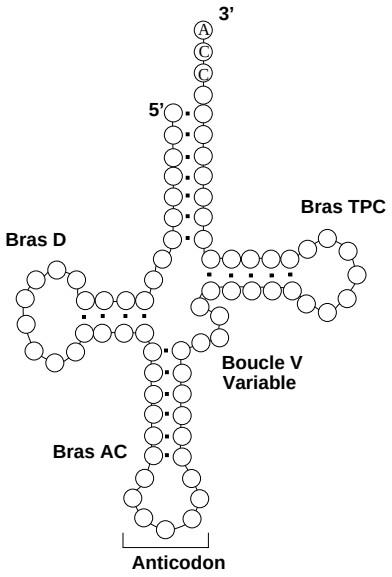


Fig. 6.19: Structure secondaire d'un ARNt de levure

Les ARN de transfert (ARNt)

Ils représentent environ 15% des ARN cellulaires. Leur structure primaire est très variable.
Rôle : Les ARNt sont des molécules qui traduisent les codons des ARNm en acides aminés. De structures fortement contraintes et quasi identiques, ils sont caractérisés par leurs anticodons(voir Figure 6.19). Un enzyme spécifique à chaque anticodon, l'aminoacyl-ARNt synthétase, se charge

[10] La fiabilité des structures prédites *in silico* étant, pour l'instant, sujette à caution ...

d'adjoindre à chaque ARNt l'acide aminé correspondant au complémentaire de l'anticodon. Cet acide aminé est collé sur le brin 3' de l'ARNt au niveau d'un triplet CCA. Lors de la traduction, l'anticodon va s'apparier avec le codon de l'ARNm et permettre la concaténation d'un acide aminé adéquat.

Structure : Comme l'illustre la Figure 6.19, la structure secondaire de l'ARNt est très fortement contrainte. Sa forme de trèfle est invariable, et seule la dimension du renflement(Boucle V de la Figure6.19) permet de distinguer les structures secondaires de différents ARNt.

Remarque : On rencontre dans les ARNt des bases inhabituelles chez les ARN (pseudouridine, inosine ou encore des bases méthylées) qui sont en fait des bases modifiées après la transcription afin de conserver la structure tridimensionnelle de l'ARNt.

Les ARN ribosomiques (ARNr)

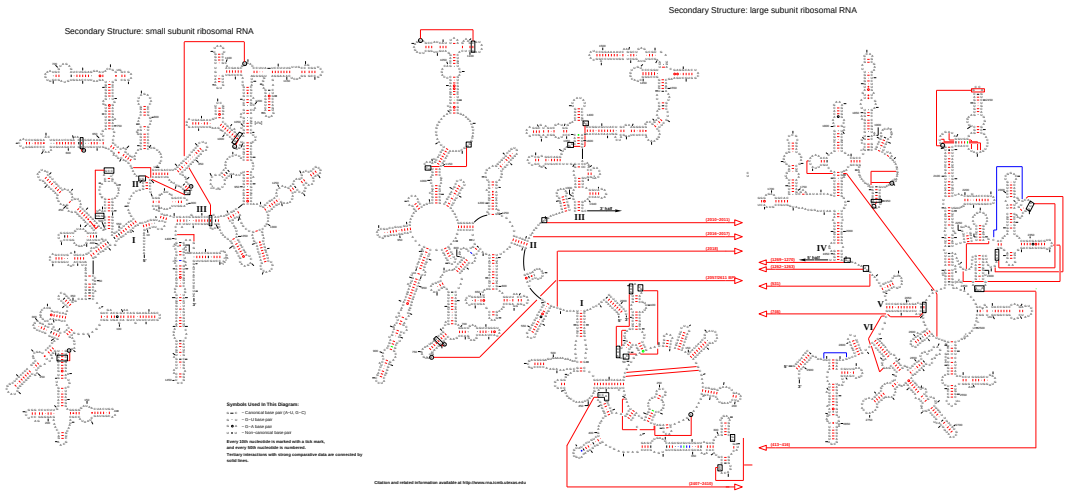


Fig. 6.20: Structure secondaires des ARN ribosomaux 16S (gauche) et 23S (droite) chez *Escherichia coli* établis par une approche comparative[22]

Ils représentent près de 80% des ARN cellulaires.

Rôle : Ils forment, avec des protéines spécialisées, le ribosome, qui est le siège de la traduction. Ils sont caractérisés par leur constante de sédimentation, exprimée selon une unité S(de Svedberg).

Structure : Les structure secondaire classiquement associées aux ARNr sont assez variées, même au sein d'une classe d'ARNr ayant même constante de sédimentation. On constate la présence de pseudo-noeuds, dont certains sont impliqués dans des tetraboucles GNRA[11], motifs structurels parfois non-contigus et essentiels à la stabilisation des complexes impliquant des ARN (ribosome, par exemple).

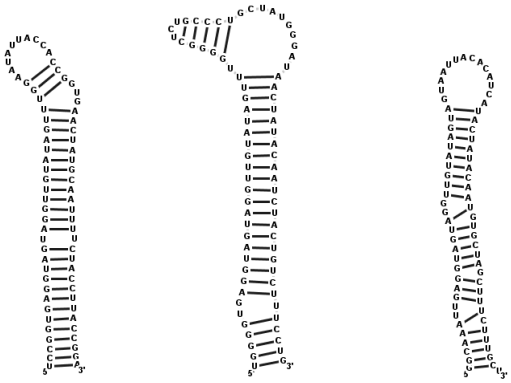
[11] Où N est un raccourci pour une base non-contrainte, et R représente une base purique (A ou G).

Famille	*Localisation*	**Nom**	Taille
Procaryotes(70*S*)	*Grande Sous-Unité*(50*S*)	**5S**	120 nt
		23S	2900 nt
	Petite Sous-Unité(30*S*)	**16S**	1542 nt
Eucaryotes(80*S*)	*Grande Sous-Unité*(60*S*)	**5S**	120 nt
		5,8S	160 nt
		28S	4800 nt
	Petite Sous-Unité(40*S*)	**18S**	1900 nt

Fig. 6.21: Les différents ARNr

Micro ARN (ARNmi) et petits ARN interférents (ARNsi)

Rôle : Les ARNmi et ARNsi sont des petits ARN non-codants pour des protéines, de tailles de l'ordre de la vingtaine de nucléotides. Principalement, leur rôle est la régulation de l'expression d'un gène, par fixation sur l'ARN messager correspondant, ce qui rend plus ou moins difficile sa transcription en protéine. Ces deux classes divergent par leur origine, principalement exogène (injection expérimentale mais parfois issus d'exons) pour les ARNsi, et entièrement endogène (encodés par des gènes spécifiques) dans le cas des ARNmi. Dans le cas des ARNsi l'efficacité de la régulation du gène-cible repose sur une complémentarité presque parfaite entre un brin de l'ARNsi et l'ARNm visé. Il est alors théoriquement possible de construire des ARNmi adoptant spontanément la bonne conformation, et ciblant avec une spécificité remarquable un ARNm donné (impliqué dans une pathologie). C'est la technique d'ARN interférence, sur laquelle reposent d'immenses espoirs thérapeutiques.
Structure : Les ARNsi sont injectés directement sous la forme d'un ARN double-brin, c'est à dire de deux ARN simple-brin parfaitement complémentaires et appariés. La structure des ARNmi est quand à elle assimilable à une tige-boucle, pouvant faire apparaître des renflements ou des boucles internes, comme l'illustre la figure 6.22. La détection automatique des ARNmi au sein d'un génome est un des challenge actuel de la bioinformatique, et leur modélisation pour l'évaluation des candidats susceptible de tirer partie de formalismes syntaxiques expressifs.



C.Elegans H. Sapiens chr22 D. Melanogaster

Fig. 6.22: Trois exemples de précurseurs d'ARNmi [70]

7. GÉNÉRATION ALÉATOIRE DE STRUCTURES D'ARN *RÉALISTES*

7.1 Introduction

7.1.1 Motivation

On s'intéresse ici à la génération aléatoire de structures secondaires d'ARN. Une telle génération connaît divers champs d'applications. Tout d'abord, elle permet l'évaluation *expérimentale* de la probabilité d'apparition d'un motif structurel donné dans un ARN. Elle permet aussi d'établir des seuils pour les algorithmes de comparaisons de structures d'ARN, algorithmes faisant actuellement l'objet d'une attention soutenue de la part de la communauté bioinformatique [105; 51; 46; 3; 69; 98], tant pour la beauté intrinsèque du problème que pour ses applications multiples (Calcul de scores de similarité, recherche d'ARN non-codants dans un ADN ou encore extraction d'une sous-structure commune). Elle permettrait enfin de calibrer les algorithmes dédiés à la détection de motifs structuraux particuliers au sein d'une séquence d'ARN, comme RNAMOTIF [60] ou ERPIN [44].

L'approche classique pour la génération aléatoire de structures d'ARN tire partie d'une bijection entre un langage proche des mots de Motzkin et une variété de cartes planaires décrivant les structures d'ARN. Cependant, une implémentation uniforme de cette approche, bien qu'efficace,

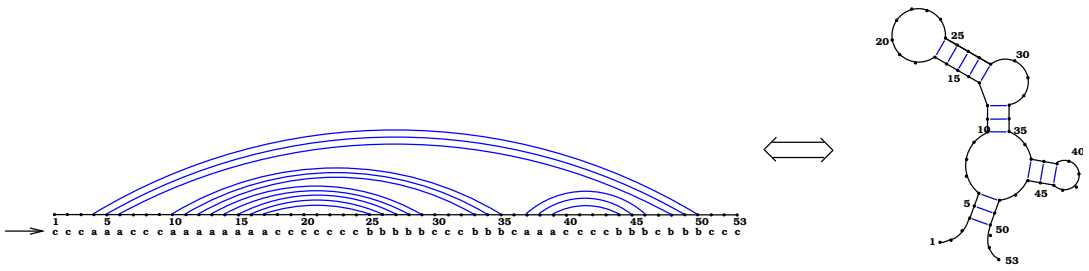


Fig. 7.1: Bijection entre les mots de Motzkin sans motif ab et structures secondaires d'ARN.

possède la désagréable propriété d'engendrer des structures à la granularité bien trop prononcée, comme l'illustre la figure 7.2. Cette apparence, plus proche du *bonsaï* que de l'arborescence épanouie, est expliquée directement par les propriétés asymptotiques du modèle uniforme. En effet, des études séparément menées par M. Nebel [64] et l'auteur de cette thèse [73] ont permis de dériver les propriétés suivantes pour les sous-structures d'ARN ribosomal, résumées dans la table 7.3. On peut les comparer à celles observées sur des ARN ribosomiques issus d'une méthode comparative, que j'ai compilées en 2003 à partir de la base de données du Gutell Lab [22], et

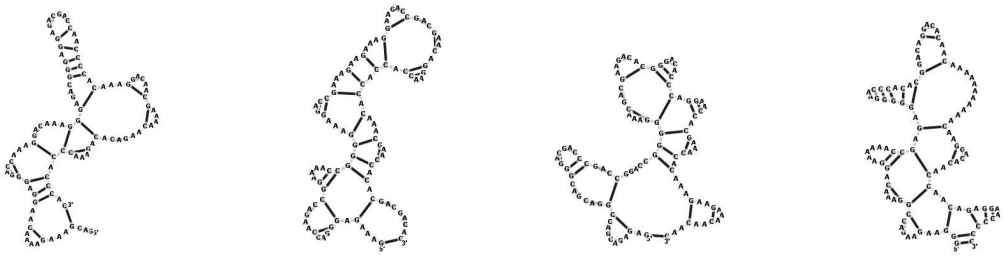


Fig. 7.2: Structures secondaires aléatoires : Exemples issus du modèle uniforme

récapitulées dans la table 7.4. L'incapacité constatée du modèle uniforme à modéliser correctement les structures d'ARN suggère alors l'emploi de contraintes additionnelles, ou l'altération de la distribution des séquences. Ding et Lawrence [33] étudient le problème de la génération de structures à partir d'une séquence. Il cherchent à engendrer un ensemble de structures tirées selon des probabilités liées au modèle d'énergie libre de Turner. Ils en déduisent un algorithme en $\mathcal{O}(n^2)$ après un précalcul en $\mathcal{O}(n^3)$ (Approche récursive). Cette complexité en $\mathcal{O}(n^2)$ de la phase de génération peut être optimisée en $\mathcal{O}(n)$ par une stratégie *boustrophédon*. Cependant leur approche se limite à une séquence, ce qui empêche l'évaluation de scores statistiques. De plus, leur attachement à un modèle d'énergie libre additif peut représenter une limite dans le temps de la validité de leur travail.
Dans son mémoire de DEA [102], F. Weinberg présente une version des grammaires non-contextuelles, pondérées au niveau des dérivations, dans un formalisme qui ressemble alors aux grammaires stochastiques. Il propose alors des pistes pour apprendre les poids d'une telle grammaire, et tente l'application d'une d'entre elles aux structures secondaires d'ARN dont les proportions de bases appariées/non-appariées sont fixées. Les résultats sont partiels, mais prometteurs.

7.1.2 Méthode

Dans ce chapitre, nous présentons une modélisation de la structure secondaire de l'ARN par des grammaires pondérées, et son application à la génération aléatoire de structures d'ARN réalistes. Par *réalistes*, nous entendons *qui possèdent des valeurs pour certains paramètres structurels en moyenne égales à celles observées dans des structures réelles*. Les paramètres structurels auxquels nous allons nous intéresser sont les nombres et tailles des différents types de sous-structures, présentées dans la figure 7.5 :

- **(H) Hélice** : Il s'agit d'une séquence de paires (Liaisons Hydrogènes) de bases empilées (Liaison Phosphodiester). Nous nous bornons ici à des considérations uniquement topologiques, on ne fait donc pas d'hypothèses sur le type de ces liaisons (Watson-Crick, Wobble ou non-canoniques). Alternativement, on peut dire qu'il s'agit des portions double-brin de l'ARN.

Distributions des bases dans les sous-structures		
Sous Structure	Taille cumulée	
Extrémité 5'	$\frac{\sqrt{5}-1}{2}+\mathcal{O}(\frac{1}{n})$	
Extrémité 3'	$\frac{\sqrt{5}-1}{2}+\mathcal{O}(\frac{1}{n})$	
Hélice brin 5'	$\frac{(5-\sqrt{5})n}{20}-\frac{3+2\sqrt{5}}{20}+\mathcal{O}(\frac{1}{n})$	
Hélice brin 3'	$\frac{(5-\sqrt{5})n}{20}-\frac{3+2\sqrt{5}}{20}+\mathcal{O}(\frac{1}{n})$	
Renflement	$\frac{(25-11\sqrt{5})n}{5}+\frac{\sqrt{5}-1}{20}+\mathcal{O}(\frac{1}{n})$	
Boucle Interne	$\frac{2(9\sqrt{5}-20)n}{5}+\frac{53-23\sqrt{5}}{20}+\mathcal{O}(\frac{1}{n})$	
Boucle	$\frac{(3\sqrt{5}-5)n}{10}+\frac{\sqrt{5}+9}{20}+\mathcal{O}(\frac{1}{n})$	
MultiBoucle	$\frac{(7-3\sqrt{5})n}{2}-\frac{7-\sqrt{5}}{4}+\mathcal{O}(\frac{1}{n})$	
Nombres d'occurences et tailles des sous-structures		
Sous Structure	Nombre	Taille
Hélice	$\sqrt{5}-2n-\frac{9\sqrt{5}+23}{8}+\mathcal{O}(\frac{1}{n})$	$\frac{3\sqrt{5}}{5}+\mathcal{O}(\frac{1}{n})$
Boucle	$\frac{5-2\sqrt{5}}{5}n-\frac{3\sqrt{5}-13}{20}+\mathcal{O}(\frac{1}{n})$	$\frac{1+\sqrt{5}}{2}+\mathcal{O}(\frac{1}{n})$
Renflement	$\frac{18\sqrt{5}-40}{5}n+\frac{57\sqrt{5}-127}{20}+\mathcal{O}(\frac{1}{n})$	$\frac{1+\sqrt{5}}{2}+\mathcal{O}(\frac{1}{n})$
Boucle Interne	$\frac{65-29\sqrt{5}}{10}n+\frac{63-28\sqrt{5}}{20}+\mathcal{O}(\frac{1}{n})$	$1+\sqrt{5}+\mathcal{O}(\frac{1}{n})$
MultiBoucle	$\frac{7\sqrt{5}-15}{10}n+\frac{69\sqrt{5}-129}{160}+\mathcal{O}(\frac{1}{n})$	$\sqrt{5}+\mathcal{O}(\frac{1}{n})$
Intervalle Intervalle par Multiboucle	$(\sqrt{5}-2)n-\frac{2+\sqrt{5}}{2}+\mathcal{O}(\frac{1}{n})$ $\frac{5+\sqrt{5}}{2}+\mathcal{O}(\frac{1}{n})$	$\frac{\sqrt{5}-1}{2}+\mathcal{O}(\frac{1}{n})$

Fig. 7.3: Propriétés asymptotiques du modèle uniforme pour les structures secondaires d'ARN

(A) Règne:CS	NB	#Bases	%Ex3+Ex5	%H	%R	%B	%I	%M
Archae 5S	1	123	4,88	61,79	3,25	13,82	10,57	4,88
16S	18	1486	1,07	60,06	4,24	11,85	9,84	12,87
23S	12	2946	0,37	56,20	3,03	13,25	13,78	13,34
Bacteria 5S	24	120	1,76	64,16	2,49	13,95	9,32	7,49
16S	231	1531	1,44	58,97	4,35	11,59	10,88	12,70
23S	100	3325	0,23	49,28	2,48	11,80	11,43	24,74
Eucaryotes 5S	1	119	0,84	62,18	1,68	13,45	15,97	5,04
16S	38	1648	0,92	52,93	4,09	12,26	11,31	18,43
23S	56	2261	0,59	50,56	2,82	16,94	11,06	18,01
Théorique (Uniforme)	-	$\rightarrow\infty$	0	55,27	8,06	17,08	4,98	14,58

(B) Règne:CS	NB	#Bases	H	R	B	I	M	#K/M
Archae 5S	1	123	9,50	1,33	8,50	6,50	6,00	3,00
16S	18	1486	9,45	2,62	5,87	5,75	9,11	2,81
23S	12	2946	8,16	1,93	5,59	7,53	11,79	4,05
Bacteria 5S	24	120	10,90	1,49	8,42	5,43	9,04	3,00
16S	231	1531	8,71	2,35	5,72	5,92	8,82	2,82
23S	100	3325	8,25	1,98	5,72	6,82	25,41	4,09
Eucaryotes 5S	1	119	12,33	2,00	8,00	9,50	6,00	3,00
16S	38	1648	8,80	2,52	6,98	6,70	14,46	2,77
23S	56	2261	8,21	2,12	7,31	7,71	14,44	3,19
Théorique (Uniforme)	-	$\rightarrow\infty$	2	1,618	1,618	3,236	2,236	3,618

Fig. 7.4: Pourcentages de bases dans les différentes sous structures **(A)**, et tailles moyennes de ces sous-structures **(B)** chez des ARN ribosomiques obtenus par génomique comparative [22] et les structures aléatoires issues du modèle uniforme.

- **(B) Boucle Terminale** : Portion d'ARN simple-brin reliant directement les deux bases terminant une hélice.

- **(R) Renflement** : Portion d'ARN simple-brin placée entre deux hélices au niveau d'un seul des deux brins (5' ou 3').

- **(I) Boucle Interne** : Portions d'ARN simple-brin placées entre deux hélices au niveau de chacun des deux brins.

- **(M) Multiboucle** : Multiples portions d'ARN simple brin organisant l'*arrivée* d'une hélice et le *départ* d'au moins deux hélices. Cette sous-structure est responsable de l'aspect *arborescent* des structures d'ARN. On peut s'intéresser, outre le nombre global de bases non appariées qu'elle contient, à la taille moyenne des portions simple-brin séparant les départ d'hélices, et surtout au nombre moyen d'hélices dont elle voit le départ. On nomme **Intervalle (K)** la portion simple-brin séparant deux hélices. Le nombre d'intervalles dans une multiboucle sera donc égal au nombre d'hélices qui en partent, moins un.

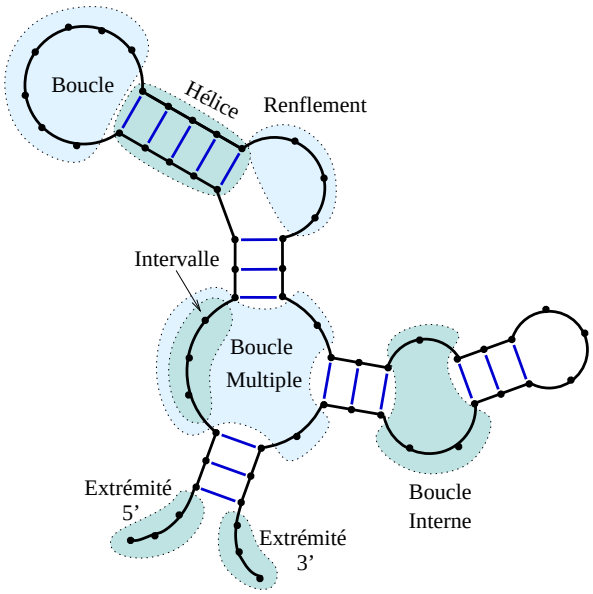


Fig. 7.5: Les différents types de sous-structures dans un ARN.

- **(Ex5,Ex3) Extrémité** (5' ou 3') : Portions d'ARN simple-brin placées aux extrémités de l'ARN.

Dans un premier temps, nous présenterons différentes grammaires équivalentes, en ce sens qu'elles permettent toutes la description de toutes les structures secondaires d'ARN. Nous discuterons leur simplicité (nombre de symboles non-terminaux), et leur capacité à discriminer les différents types de sous-structures décrites ci-dessus. Nous nous intéresserons ensuite à l'adjonction de pondérations à ces grammaires, selon le paradigme des grammaires pondérées, pour modéliser la structures de certains ARN. Dans le cas simple où l'on ne s'intéresse qu'aux proportions de bases non-appariées et appariées dans les structures engendrées, une solution analytique est dérivée grâce à une application d'une variante du théorème de Drmota. Dans des cas plus généraux, cette méthode échoue car nécessitant la résolution de systèmes d'équations bien trop complexes pour une utilisation standard d'un logiciel de calcul symbolique. On utilise alors une approche d'optimisation décrite en section 4.4.3 pour dériver des modèles pondérés pour 3 ARN : L'ARN de transfert, l'ARN ribosomal 5S bactérien et l'ARN ribosomal 23S eucaryote. On présentera en outre un modèle pour le langage des tiges-boucles, qui possède des propriétés théoriques intéressantes.

7.2 Grammaires pour la structure secondaire d'ARN

Nous présentons ici différentes grammaires pour la structure secondaire d'ARN, présentant des potentiels de personnalisation et des complexités différents. On trouvera en annexe un argument de preuve d'équivalence des langages engendrés par ces grammaires avec les structures

secondaires d'ARN telles que définies par Waterman dans [100]. Dans ces grammaires, on utilisera des symboles a et b pour représenter les bases appariées respectivement en 5' et en 3', et le symbole c pour représenter les bases non-appariées.

7.2.1 Grammaire de Nebel[65]

Dans [65], M. Nebel propose une jolie grammaire pour les structures secondaires d'ARN.

$$\begin{aligned} R &\rightarrow T \mid TR \\ T &\rightarrow c \mid aRb \end{aligned}$$

Cette grammaire présente la propriété de ne pas posséder de transition $N \rightarrow \varepsilon$. Cependant, elle ne permet pas le marquage simple des sous-structures, ni l'adjonction de tailles minimale pour les différentes sous-structures. En effet, l'initiation et la prolongation d'une sous structure sont pris en charge par le même non-terminal (T).

7.2.2 Grammaire inspirée des mots de Motzkin

Une façon de caractériser les structures secondaires d'ARN en tant que langage consiste à dire qu'il s'agit de mots de Motzkin sans motif ab. La grammaire ci-dessous, basée sur celle des mots de Motzkin, distingue les bases issues d'un non-terminal *encadrée* par deux bases appariées (T) des autres (S).

$$\begin{aligned} S &\rightarrow aTbS \mid cS \mid \varepsilon \\ T &\rightarrow aTbS \mid cS \end{aligned}$$

Cette grammaire permet la distinction des bases appariées (a-b) responsables de l'initiation d'une hélice (S) de celles prolongeant une hélice existante (T). Cela permet, outre le marquage par un symbole m_H des débuts d'hélices pour capturer le nombre d'hélices dans un ARN, d'injecter une taille minimale h pour les hélices. De plus, en remarquant que le non-terminal T engendre des portions de la structure directement encadrées par des bases appariées, on peut alors injecter une taille minimale b dans les boucles terminales, ainsi qu'un marquage de celles-ci par un symbole terminal dédié m_B.

$$\begin{aligned} S &\rightarrow m_H a^{h-1} T b^h S \mid cS \mid \varepsilon \\ T &\rightarrow aTbS \mid cT \mid m_B c^{b-1} \end{aligned}$$

Mais cette grammaire ne permet pas encore d'imposer des contraintes sur les portions simple-brin. En particulier, il est impossible d'attribuer le c de la réécriture $T \rightarrow cS$ à une sous-structure simple-brin précise.

7.2.3 *Grammaire complète : marquage des sous-structures [73; 65]*

Cette dernière décomposition, qui a été découverte indépendamment par Nebel [65] et l'auteur du présent document [73], repose sur l'arborescence sous-jacente à une structure d'ARN.

$$\begin{array}{rcl} X & \rightarrow & NH_iN \mid B \mid M \\ N & \rightarrow & cN \mid \varepsilon \\ H_i & \rightarrow & aHb \\ H & \rightarrow & cNH_i \mid H_iNc \mid cNH_iNc \mid aHb \mid B \mid M \\ B & \rightarrow & cN \\ M & \rightarrow & NH_iM \mid NH_iNH_iN \end{array}$$

On peut désormais distinguer les paires de bases initiant des échelles (H_i) de celles les prolongeant (H). On peut aussi distinguer les boucles, les renflements et les boucles internes (resp. B, productions cNH_i/H_iNc et cNH_iNc). Les multiboucles peuvent aussi faire l'objet d'un marquage spécifique au sein du non-terminal M et à chacune de ses occurrences dans le reste de la grammaire. Les contraintes de tailles minimales pour les hélices (h) et les boucles (b) peuvent en outre être injectées, d'où l'apparition de termes de la forme α^k dans la grammaire, qui correspondent à la concaténation de k copies de α. Il en résulte la grammaire suivante :

$$\begin{array}{rcl} X & \rightarrow & NH_iN \mid B \mid M \\ N & \rightarrow & K \mid \varepsilon \\ H_i & \rightarrow & m_H a^{h-1} H b^h \\ H & \rightarrow & RH_i \mid H_iR \mid m_I IH_iIf \mid aHb \mid B \mid M \\ B & \rightarrow & dB \mid m_B d^{b-1} \\ R & \rightarrow & eR \mid m_R \\ I & \rightarrow & fI \mid \varepsilon \\ M & \rightarrow & Nm_M a^{h-1} Hb^{h-1} m_K M \mid Nm_H a^{h-1} Hb^{h-1} m_K Nm_H a^{h-1} Hb^{h-1} m_K N \\ K & \rightarrow & gK \mid \varepsilon \end{array}$$

Remarque : L'application à la grammaire du morphisme ϕ tel que

$$\begin{array}{rclrcl} \Phi(d) & = & c & \Phi(m_H) & = & a \\ \Phi(e) & = & c & \Phi(m_I) & = & c \\ \Phi(f) & = & c & \Phi(m_B) & = & c \\ \Phi(g) & = & c & \Phi(m_M) & = & a \\ & & & \Phi(m_K) = b & & \end{array}$$

nous permet, en conjonction avec des valeurs pour les minimums $h = 1$ et $b = 1$, de retrouver une grammaire pour les mots de Motzkin sans facteur ab.

Ces marquages permettent de mettre en relation les nombres d'occurrences des différents symboles terminaux et les nombres et tailles des différentes sous-structures dans les structures d'ARN engendrées.

Type de sous-structure	Nombre total de bases du type
Hélices	$\#a+\#b+\#m_H+\#m_M+\#m_K$
Boucles	$\#d+\#m_B$
Boucles Internes	$\#f+\#m_I$
Renflements	$\#e+\#m_R$
MultiBoucles	$\#g$
Intervalle	$\#g$
Type Sous-structures	Nombre de sous-structure du type
Hélices	$\#m_H+\#m_M$
Boucles	$\#m_B$
Boucles Internes	$\#m_I$
Renflements	$\#m_R$
MultiBoucles	$\#m_M$
Intervalles	$\#m_K+\#m_M$

7.3 Cas analytiques simples : Les proportions de bases appariées

7.3.1 Les tiges-boucles

La structure de tige-boucle découle d'une autre façon de décomposer la structure de l'ARN que celle utilisée tout au long de ce chapitre. Elle débute par une hélice et se prolonge jusqu'à qu'une boucle terminale soit rencontrée. Des renflements et autres boucles internes peuvent être comprises dans cette structure, mais pas de multiboucle. Ces structures peuvent être modélisées par la grammaire $\mathcal{G}_P$ suivante :

$$\begin{aligned} P &\rightarrow RaPbR \\ P &\rightarrow \varepsilon \\ R &\rightarrow cR \\ R &\rightarrow \varepsilon \end{aligned}$$

On peut déjà remarquer que, bien que ce langage soit algébrique, car ressemblant furieusement au langage des palindromes, sa série génératrice sera rationnelle, ce qui exclut de fait l'utilisation du type de technique (Théorème de Drmota) utilisées pour contraindre la proportion de bases appariées dans l'ARN. Cette rationalité est due au fait que l'algébricité stricte de ce langage tient à la non-commutativité du produit de concaténation, propriété non-préservée par la transposition dans l'univers des séries génératrices. Cependant, les comportements asymptotiques des coefficients des séries rationnelles étant mieux connus, on dérive le résultat suivant.

Théorème 10 :
Soient $\pi_{\{a,b\}}$ le poids associé aux symboles appariés a et b, et π_c le poids associé au symbole non-apparié c, alors les proportions asymptotiques $\mu_{\{a+b\}}$ (resp. μ_c) de bases appariées (resp. non-appariées) dans les séquences engendrées à partir de la grammaire $\mathcal{G}_P$ des tiges-boucles, pondérée par π sont données par

$$\mu_{\{a+b\}} = \frac{\pi_{\{a,b\}}}{\pi_{\{a,b\}}+\pi_c} \qquad \mu_c = \frac{\pi_c}{\pi_{\{a,b\}}+\pi_c}$$

PREUVE : On s'intéressera uniquement à la proportion μ_c de symboles c, son complément à 1, $\mu_{\{a+b\}}$, étant directement déductible. De plus, on se ramènera sans perte de généralité à l'étude d'une pondération π telle que $\pi_{\{a,b\}} = 1$ et π_c rationnel. En effet, si l'on souhaite adapter le résultat à π' tel que $\pi'_{\{a,b\}} \neq 1$, alors il suffit de remarquer que l'impact des pondérations est stable par homothétie, et qu'on peut donc toujours se ramener à l'étude du cas $\pi_{\{a,b\}} = 1$ et $\pi_c = \frac{\pi'_c}{\pi'_{\{a,b\}}}$.

Une fois ces quelques simplifications, transparentes, du problème effectuées, on transpose la grammaire en système d'équations sur les séries génératrices pondérées associées aux non-terminaux :

$$\begin{cases} P(z,u) &= R(z,u)^2 z^2 S + 1 \\ R(z,u) &= \pi_c u z R(z,u) + 1 \end{cases}$$

En résolvant ce système, on obtient alors

$$\begin{aligned} P(z,u) &= \frac{(1-\pi_c u z)^2}{(\pi_c u z + z - 1)(\pi_c u z - z - 1)} \\ P(z,1) &= \frac{\pi_c^2}{\pi_c^2 - 1} + \frac{1}{(2\pi_c + 2)} \frac{1}{(1 - z(\pi_c + 1))} - \frac{1}{(2\pi_c - 2)} \frac{1}{(1 - z(\pi_c - 1))} \end{aligned}$$

Le terme dominant dans cette décomposition est le deuxième élément simple, dont le comportement asymptotique nous donne :

$$[z^n]P(z,1) \sim \frac{(\pi_c + 1)^n}{2\pi_c + 2} + \mathcal{O}((\pi_c - 1)^n)$$

De plus,

$$\frac{\partial P(z,u)}{\partial u}(z,1) = \frac{2\pi_c z^3 (1 - \pi_c z)}{(1 - z(\pi_c + 1))^2 (1 - z(\pi_c - 1))^2}$$

Sans rentrer dans les détails du calcul, une décomposition en éléments simples désigne comme terme dominant la fraction rationnelle $m(z)$ telle que

$$m(z) = \frac{\pi_c}{2(\pi_c + 1)^2 (1 - z(\pi_c + 1))^2}$$

et dont le comportement des coefficients est tel que :

$$\begin{aligned} & [z^n]m(z) = \frac{\pi_c}{2(\pi_c + 1)^2} (\pi_c + 1)^n (n + 1) \\ \Rightarrow \quad & [z^n]\frac{\partial P(z,u)}{\partial u}(z,1) \sim \frac{\pi_c}{2(\pi_c + 1)^2} (\pi_c + 1)^n n + \mathcal{O}((\pi_c + 1)^n) \end{aligned}$$

■

Donc, en se remémorant la relation $\mu_c = \frac{[z^n]\frac{\partial P(z,u)}{\partial u}(z,1)}{n[z^n]P(z,1)}$ entre une série génératrice $P(z,u)$ de langage et la fréquence μ_c d'un terminal c, on obtient

$$\mu_c = \frac{\pi_c}{\pi_c + 1}$$

En se souvenant que $\pi_c = \frac{\pi'_c}{\pi'_{\{a,b\}}}$ dans un système de pondération π' équivalent car homothétique, on trouve

$$\mu_c = \frac{\frac{\pi'_c}{\pi'_{\{a,b\}}}}{\frac{\pi'_c}{\pi'_{\{a,b\}}} + 1} = \frac{\pi'_c}{\pi'_{\{a,b\}} + \pi'_c}$$

7.3.2 La structure secondaire d'ARN

Considérons la grammaire de la section 7.2.2 pour les structures secondaires d'ARN, au sens de Waterman [100] : On s'intéresse à un système de pondérations permettant de contraindre la proportion d'occurrences de la lettre c, qui représente une base non-appariée.

Théorème 11 :
Le poids π_c associé aux bases non-appariées afin de réaliser une proportion μ_c de bases non-appariées est tel que :

$$\pi_c = \frac{4\mu_c^2}{1-\mu_c^2} \tag{7.1}$$

PREUVE : En transformant les règles de la grammaire en équations algébriques sur les séries liées aux non-terminaux, on remarque que $T \equiv S - 1$. On peut donc se limiter à l'étude d'un système à deux variables fonctionnelles S et x :

$$\begin{cases} S &= x(S-1)xS + xu_c\pi_c S + 1 \\ 0 &= 1 - x^2 S - x(S-1) - \pi_c u_c x \end{cases}$$

En résolvant ce système de façon à ce que $x(1) \geq 0$, on obtient

$$x(u_c) = \frac{\pi_c u_c + 2 + \sqrt{\pi_c^2 u_c^2 + 4\pi_c u_c}}{2}$$

et

$$\mu_c = \frac{1}{x(1)} \frac{\partial x(u_c)}{\partial u_c}(1) = \frac{\pi_c}{\sqrt{\pi_c(\pi_c+4)}}$$

ce qui, en inversant l'équation, nous donne l'équation (7.1). ∎

Par exemple, si l'on souhaite 50% de bases non appariées dans un ARN engendré aléatoirement, il suffit de fixer $\pi_a = \pi_b = 1$ et $\pi_c = \frac{4}{3}$.

Une telle approche ne supporte malheureusement pas le *passage à l'échelle* que représente l'adjonction à la grammaire de paramètres additionnels. Les capacités de résolution d'un logiciel de calcul formel, dont l'emploi est toutefois nécessaire dès que plus de deux variables sont introduites, se trouve très vite dépassées. De plus, il n'est pas possible de *traiter les variables indépendamment*. On devra donc faire appel au logiciel développé à partir du couplage d'un algorithme d'évaluation des proportions des symboles dans un modèle pondéré et d'un algorithme d'optimisation pour pouvoir prendre en compte plus de paramètres (voir section 4.4.3).

7.4 Approche optimisation : Expériences

7.4.1 L'ARNt

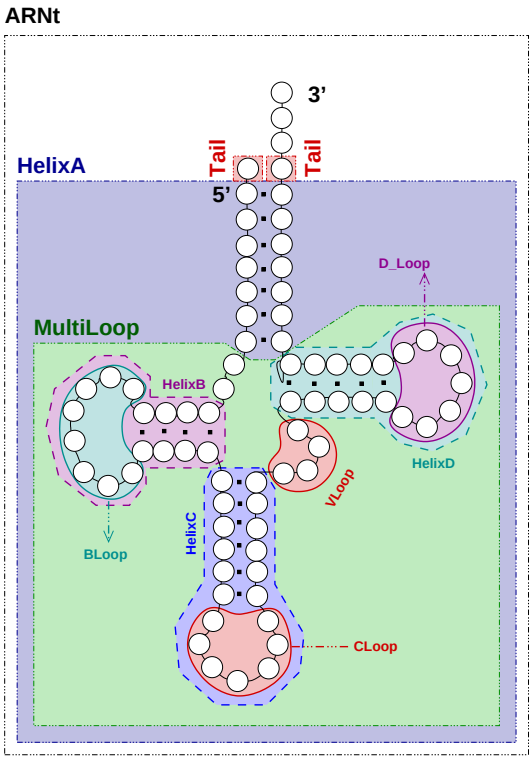


Fig. 7.6: Correspondance entre les non-terminaux de la grammaire de l'ARNt et les régions qu'ils engendrent.

On considère le modèle simple suivant pour l'ARNt, dont la structure est très contrainte comparée aux structures secondaires générales ci-dessus :

ARNt	→	Tail HelixA Tail t t t
Tail	→	t Tail | ε
HelixA	→	a HelixA a | bv bv HelixB bv HelixC VLoop HelixD
HelixB	→	b HelixB b | BLoop
HelixC	→	c HelixC c | CLoop
HelixD	→	d HelixD d | DLoop
BLoop	→	bb BLoop | bb bb bb
CLoop	→	bc CLoop | bc bc bc
DLoop	→	bd DLoop | bd bd bd
VLoop	→	bv VLoop | bv bv bv

Les relations entre les non-terminaux de la grammaire et les portions de l'ARNt qu'ils engendrent

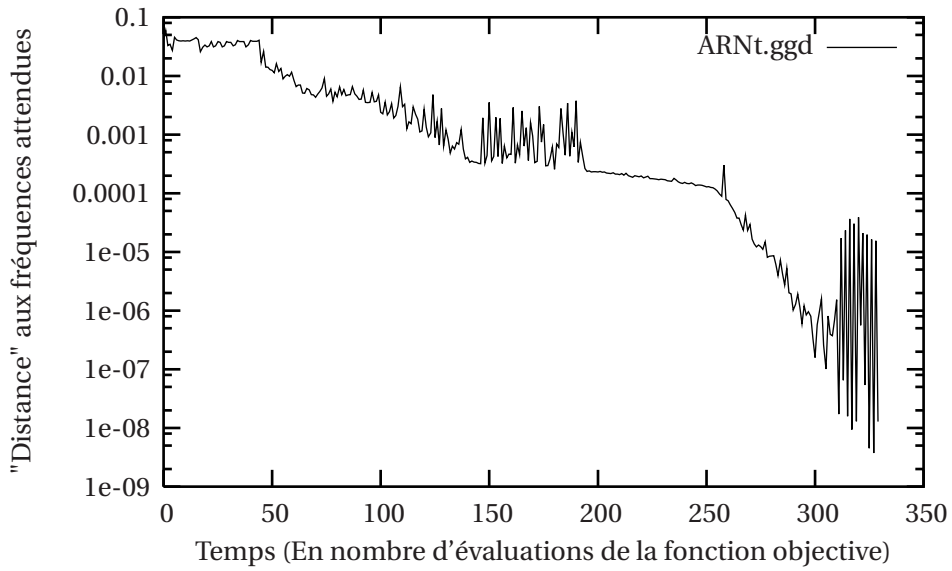


Fig. 7.7: Evolution de la distance aux fréquences souhaitées en fonction du temps (Échelle logarithmique) pour un modèle simple d'ARNt.

sont illustrées par la figure 7.6. On se fixe comme but les tailles moyennes finales suivantes pour les différentes hélices et boucles :

HelixA	7 paires	BLoop	12 bases	Tail	5 bases
HelixB	4 paires	CLoop	7 bases		
HelixC	4 paires	DLoop	7 bases		
HelixD	4 paires	VLoop	11 bases		

On répercute alors ces contraintes sur les nombres d'occurrence des symboles du vocabulaire terminal :

Symbole	Nombre d'occurences	Symbole	Nombre d'occurences
a	14	bb	12
b	8	bc	7
c	8	bd	7
d	8	bv	11
t	5		

On exécute alors notre algorithme d'optimisation, décrit en section 4.4.3 page 59, et on obtient le tracé de la figure 7.7 pour l'évolution de la distance aux fréquences attendues réalisées par les propositions de l'algorithmes à travers le temps. Il est à noter que la meilleure proposition de l'algorithme induit des fréquences estimées à une distance euclidienne égale à $3.733597\ldots 10^{-9}$ des fréquences attendues. On croit déceler dans ce graphique une convergence exponentielle,

c'est à dire que le temps nécessaire pour estimer des poids réalisant des fréquences à une distance de l'ordre de 10^{-k} de l'objectif, prendrait un temps proportionnel à k.

7.4.2 ARNr 5S et 23S

Nous nous intéressons maintenant à l'application aux ARNr du calcul des pondérations, en basant notre approche sur la grammaire générale de la section 7.2.3. Ces ARN étant réputés jouer un rôle de part leur structure, celle-ci a donc fait l'objet d'études, et de nombreuses données sont disponibles. Elles permettent l'évaluation des différents paramètres structurels résumés dans la figure 7.4.

Règne:CS	NB	#Bases	%Ex3+Ex5	%H	%R	%B	%I	%M
Bacteria 5S	24	120	1,76	64,16	2,49	13,95	9,32	7,49
Eucaryotes 23S	56	2261	0,59	50,56	2,82	16,94	11,06	18,01

Règne:CS	NB	#Bases	H	R	B	I	M	#K/M
Bacteria 5S	24	120	10,90	1,49	8,42	5,43	9,04	3,00
Eucaryotes 23S	56	2261	8,21	2,12	7,31	7,71	14,44	3,19

On s'intéressera plus particulièrement au 5S bactérien et au 23S eucaryote, dont les paramètres sont rappelés ci-dessus. Ils possèdent des proportions de bases appariées assez différents. De plus, les nombres de bases non-appariées dans les multiboucles sont très différents, suggérant l'existence de nombreux pseudo-noeuds, éliminés par l'algorithme d'évaluation des paramètres structurels, dans les multiboucles de 23S. Le problème d'optimisation associé au calcul des pondérations dépend *critiquement* du nombre de paramètres structurels distincts, c'est pourquoi on va regrouper certains paramètres, donnant naissance aux deux modèles suivants :

- Modèle *hélicoïdal* : Il s'agit de contraindre les nombres et tailles de hélices. Cela revient à uniquement considérer les occurrence des symboles a, c et m_H dans la grammaire de la section 7.2.2.

- Modèle *simplifié* : On part de la grammaire la plus générale (section 7.2.3, page 104), qui fait apparaître 11 paramètres structurels. Parmi ceux ci, certains jouent des rôles qu'on peut assimiler sans trop perdre d'information. En particulier, le nombre total de bases dans des renflements, des boucles internes et des multiboucles sera considéré globalement, de même que les nombres de renflements et de boucles internes. On se ramène donc à un modèle à 8 paramètres, dont la résolution est en temps réaliste.

On exécute alors l'algorithme d'optimisation sur ce modèle, et on trouve les résultats illustrés par la figure 7.8. Dans le modèle *hélicoïdal*, une convergence brutale arrive très vite, permettant l'obtention de poids réalisant des fréquences à une *distance* 10^{-20} de celles souhaitées.
Dans le cas *simplifié* à 8 paramètres, il est difficile de prédire le type de convergence de l'algorithme, mais on peut prédire qu'une telle convergence aura lieu. Peut être est elle exponentielle de *fort* coefficient (proche de 1) ? Il est difficile de répondre à cette question en l'absence d'une étude *théorique* précise de notre fonction objectif.

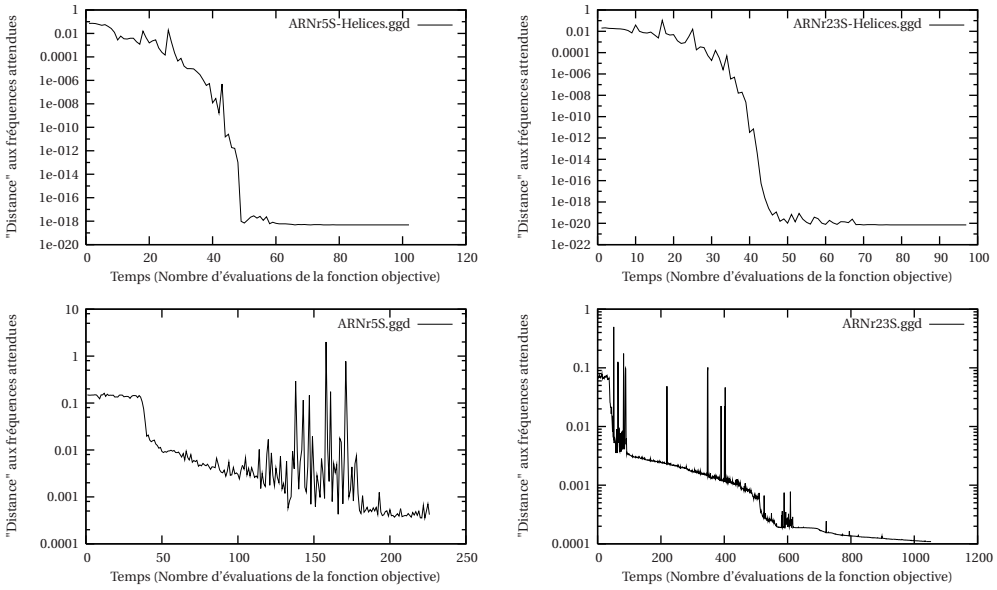


Fig. 7.8: Distance aux fréquences souhaitées au cours du temps. Modèles *hélicoïdal* (`ARNr5S-Helice.ggd` et `ARNr23S-Helice.ggd`), et *simplifié* (`ARNr5S.ggd` et `ARNr23S.ggd`).

7.5 Conclusion

On dispose, par application d'une approche d'optimisation présentée dans le chapitre précédent, d'un mécanisme d'évaluation automatique de pondérations réalisant des paramètres structuraux correspondant à un ou plusieurs ARN. A partir de ces pondérations, il est désormais possible d'engendrer des structures secondaires d'ARN *mimant* la réalité. Ce mimétisme se limite dans l'approche actuelle à des valeurs égales en moyenne pour les nombres et tailles des sous-structures, mais il est aisé d'étendre cette notion à d'autres, pourvu que ceux-ci puissent être modélisés par des grammaires non-contextuelles.

D'autre part, les observations du comportement de l'algorithme d'optimisation doivent être corroborées par une étude théorique rigoureuse des *paysages* dessinés par notre fonction objective. En général, on pourrait s'interroger sur l'*unicité du minimum local*, condition nécessaire pour assurer la convergence vers une pondération réalisant *exactement* les fréquences désirées.

Dans le cas de l'ARNt, une telle étude nous permettrait de mieux comprendre l'origine des oscillations finales, ou encore de savoir si la rapidité de la convergence de l'algorithme, frappante en présence de nombreux paramètres, est due à l'aspect essentiellement rationnel du langage. Dans le cas de l'ARNr, on peut s'interroger sur la nature et l'existence d'une convergence dans le cas à 8 paramètres et, sur les raisons de l'existence de tels régimes de convergence.

Enfin, on pourrait intégrer l'approche *optimisation* au sein d'un logiciel de calcul symbolique, de façon à tirer parti du calcul automatique de l'asymptotique des coefficients de séries algébriques [83], ce qui permettrait une évaluation plus pertinente, et en un temps constant sur n, de la fonction objectif.

8. PLANARISATION DE LA STRUCTURE D'ARN

8.1 Motivation

En de nombreuses occasions au cours de cette thèse, j'ai été amené à confronter mes résultats et expériences sur la structure de l'ARN à des données expérimentales. Celles ci peuvent être de plusieurs type : Elles peuvent soit résulter de l'application *in silico* de méthodes bioinformatiques (repliement, voir un état de l'art exhaustif dans [62] ou génomique comparative [22]), soit être obtenues par une observation biochimique des molécules d'ARN (Résonance Magnétique Nucléaire, cristallisation aux rayons X ou cryo-microscopie électronique).

Les premières ont l'avantage d'être disponibles en grande quantité dans des bases de données, ce qui peut leur conférer une valeur statistique. Elles ont cependant l'inconvénient de dépendre fortement du modèle thermodynamique ou évolutif utilisé pour les résoudre. Il en résulte un biais mis en évidence lors de la comparaison par méthode de centroïdes [32] des structures obtenues par MFold [107] (modèle d'énergie libre de Turner) à celles issues de la génomique comparative [22]. L'existence d'un tel biais peut s'expliquer l'approximation que constituent ces modèles, qui se doivent d'être particulièrement simples afin que les temps de calcul ne soient pas trop prohibitifs. On peut aussi aussi pointer du doigt les contraintes topologiques fortes imposées aux structures obtenues. En effet, bien que des avancées récentes ont été constatées dans ce domaine [23; 87], on ne dispose toujours pas de valeurs expérimentales pour l'énergie libre des pseudo-noeuds. En outre, des problèmes liés à la théorie de la complexité [58] limitent l'univers des structures explorées par les algorithmes de repliement à des structures *planaires*, c'est à dire représentables sans croisement sur un demi-plan, ou à des sous-ensembles restreints de pseudo-noeuds [79; 77][1]. Cette contrainte, pertinente d'un point de vue algorithmique dans de nombreux cas [18; 17], est a priori sans fondement biologique.

On accorde donc souvent un plus grand degré de confiance à celles résolues expérimentalement et archivées dans des bases de données, parmi lesquelles la PDB [16], la NDB [15] ou RNABase [63]. Des méthodes algorithmiques issues de la biochimie [55; 104] permettent la déduction, à partir des coordonnées atomiques, des appariements classés selon la nomenclature Leontis/Westhof. Ces méthodes laissent entrevoir la possibilité d'un nouveau corpus de structures déterminées expérimentalement, à partir desquelles il serait désormais possible d'étalonner des outils de détection, d'évaluer les sensibilités d'algorithmes heuristiques, de connaître les complexités *pratiques* de nos outils ou encore de comparer les performances de divers algorithmes s'attachant à résoudre le même problème.

[1] On trouvera dans [24] une comparaison des espaces de conformations explorés par les différents algorithmes incluant des classes restreintes de pseudo-noeuds.

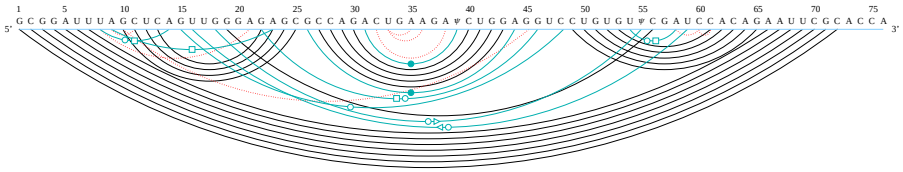


Fig. 8.1: Liaisons inférées par RNAView[104] à partir de la structure 3D d'un ARNt de phénylalaline de levure déterminée par rayons X[88]. On remarquera que même la restriction de cette structure aux liaisons canoniques Watson Crick/Watson-Crick Cis (Arcs noirs) n'est pas dessinable dans un demi-plan.

Cependant, les structures ainsi obtenues sont rarement (sinon jamais) dessinables sur un demi-plan, comme l'illustre le schéma "aplani" de la structure d'un ARNt déterminée par rayons X [88] présenté dans la figure 8.1. De nombreux autres exemples peuvent être trouvés dans la NDB [15] dont la structure, même restreinte aux liaisons canoniques WC/WC Cis (Voir [56]) n'est pas dessinable sur le demi-plan sans croisements. Or, de nombreux outils en bioinformatique ne savent efficacement travailler que sur des structures planaires, pour des raisons évoquées ci dessus.

On se propose donc d'étudier le problème de la planarisation des structures d'ARN, c'est à dire l'extraction de la *quantité maximale d'information* contenue dans un schéma similaire à la figure 8.1, telle que la structure obtenue soit dessinable sur un demi-plan. Alternativement, on pourra parler de sélection d'un sous-ensemble des appariements, optimal selon un certain critère et dessinable sans croisement sur le demi plan supérieur.

8.2 Critères d'optimalité

1. **Maximiser le nombre d'appariements :**
   On peut tout d'abord s'attacher à maximiser le nombre d'appariements. On pourra ou non choisir de se limiter aux liaisons *canoniques*, c'est à dire Watson-Crick/Watson-Crick Cis selon la nomenclature Léontis/Westhof [56]. Une telle maximisation présente l'intérêt de se transposer dans le cas des protéines dont les interactions tertiaires seraient résolues, et dans lesquelles on souhaiterait adopter une définition élargie de la structure secondaire, ne se limitant pas aux hélices-α et aux feuillets-β.

2. **Minimiser l'énergie libre de la structure :**
   Il s'agit cette fois de trouver, parmi l'ensemble des structures secondaires constituées d'un sous-ensemble des appariements initiaux, celle dont l'énergie est minimale selon un modèle thermodynamique. On pourra plus particulièrement s'intéresser au modèle de Turner, sur lequel repose MFold [106].

Remarque : Il peut paraître paradoxal de réduire à une forme planaire une structure dont la richesse réside justement dans ses interactions tertiaires. De plus, l'utilisation de l'énergie libre dans le modèle de Turner comme critère à optimiser pourrait laisser penser que le résultat d'une

exécution de l'algorithme sur une structure déterminée expérimentalement serait similaire à celui de MFold sur sa séquence. Il n'en est cependant rien car notre algorithme, en restreignant l'espace des conformations aux appariements observés, retournera un repliement presque sûrement sous-optimal relativement au modèle thermodynamique utilisé. Notre espoir est que cette restriction de l'espace de recherche permette à des phénomènes non-capturés par les modèles thermodynamiques de s'exprimer. On pourra évoquer les interactions de l'ARN avec son milieu, ou bien la séquentialité de sa génération.

8.3 Planarisation de structure secondaire maximisant le nombre d'appariements

On commence par définir une structure secondaire comme un objet strictement topologique, auquel on adjoindra des informations supplémentaires ultérieurement.

Définition 22 (Structure générale) :
Une structure générale est un graphe (V, A) *où* $V = [1, n]$ *et* $A \subseteq \{(i, j) \in [1, n]^2 | i < j\}$. □

On dit d'une structure générale $S = (V, A)$ qu'elle est inclue dans $S' = (V, A')$ ssi $A \subseteq A'$. On écrit alors $S \subseteq S'$.

Définition 23 (PLANAR-MAX-ARCS) :
Étant donnée $S = (V, A)$ *une structure générale, renvoyer* $S' = (V, A')$ *telle que :*

1. *Inclusion :* $S' \subseteq S$
2. *Planarité :*

$$\forall (i, j) \neq (i', j') \in A',\ (i \neq i') \wedge (j \neq j') \wedge \left([i, j] \cap [i', j'] = \left\{ \begin{array}{l} \varnothing \\ ou\ [i, j] \\ ou\ [i', j'] \end{array} \right. \right)$$

3. *Optimalité :* $\forall A'' \subseteq A$ *tel que* (V, A'') *planaire,* $|A''| \leq |A'|$ □

Tout d'abord, on rappellera que l'algorithme de repliement de Nussinov-Jacobson [67] se donne pour objectif de retourner la structure secondaire planaire ayant le plus grand nombre d'appariements possibles. De façon équivalente, celui ci sélectionne, dans la structure générale comprenant tous les appariements compatibles avec la séquence(voir figure 8.2), la structure planaire ayant un maximum d'appariements. C'est même le point de vue explicitement adopté dans [96], qui proposent un algorithme parallèle heuristique partant de la structure générale développée avant d'en sélectionner un sous-ensemble planaire. On se propose donc d'adapter l'algorithme de Nussinov en limitant son espace de recherche, c'est à dire en ne considérant que des appariements compris dans une structure secondaire générale paramètre de l'algorithme. Pour cela, on commencera par reformuler l'algorithme de Nussinov.

On s'appuiera pour cela sur une décomposition des structures secondaires, celle de Waterman découverte lors du dénombrement de ces structures [100] et illustrée par la figure 8.3. Contrairement à celle qui sous-tend l'algorithme de Nussinov [67], celle ci est non-ambiguë. Elle possède en outre l'avantage de regrouper dans un même *cas* l'appariement et la disjonction

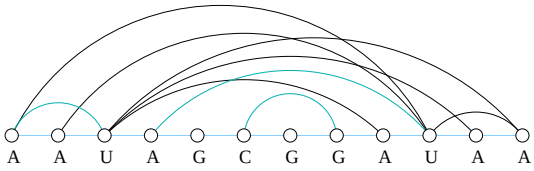


Fig. 8.2: Structure secondaire construite à partir de la séquence `AAUAGCGGAUAA`, et faisant apparaître tous les appariements potentiels `A-U` et `C-G`. L'ensemble des arcs clairs constitue l'un des repliements optimaux dans le modèle de Nussinov.

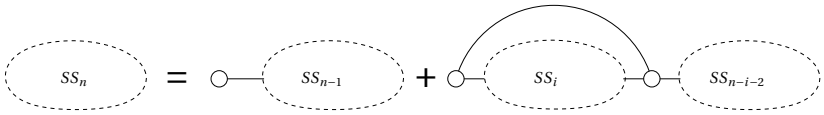


Fig. 8.3: Décomposition des structures secondaire de Waterman : *Une structure de taille n est soit une structure de taille $n-1$ précédée d'une base non-appariée, soit deux structures de tailles i et $n-i-2$, insérées respectivement sous et à droite d'un couple de bases appariées.*

en deux sous-structures *autonomes*, ce qui nous permettra d'optimiser le $\mathcal{O}(n^3)$ de l'algorithme original. On propose alors la récurrence suivante pour le calcul du nombre maximal d'appariements compris dans un sous-ensemble d'une structure secondaire donnée.

Théorème 12 :
Soit $S = (V, A)$ une structure générale, alors le nombre $\theta_{[i,j]}$ d'arcs compris dans la plus grande sous-structure $S' \subseteq S$ planaire, restreinte aux bases et arcs compris dans l'intervalle $[i, j]$, obéit à la récurrence suivante :

$$\begin{aligned} \theta_{\varnothing} &= 0 \\ \theta_{[i,j]} &= \max \begin{cases} \theta_{[i+1,j]} & (a) \\ \theta_{[i+1,k-1]} + \theta_{[k+1,j]} + 1,\ \forall (i,k) \in A, k \leq j & (b_k) \end{cases} \end{aligned}$$

où $(i > j) \Rightarrow ([i, j] \Leftrightarrow \varnothing)$ pour la simplicité des notations.

PREUVE : Par induction : On suppose que les $\theta_{(i',j')}$ sont correctement calculés par la récurrence pour tout $[i', j'] \subset [i, j]$.
Base : Le nombre maximal d'arc compris dans un intervalle vide de la structure initiale est 0.
Cas Général : Considérons la base i dans la structure optimale (V, A_{Max}), $A_{Max} \subseteq A$, il se dégage deux cas :

- Soit i est non-appariée, alors seuls les appariements de $[i+1, j]$ contribuent à $A_{Max}0$, et donc $\theta_{(i,j)} = \theta_{(i+1,j)}$, dont le calcul par la récurrence est supposé correct.

- Soit i est appariée à une base k, ce qui suppose que $(i, k) \in A$. Comme (V, A_{Max}) est planaire, A_{Max} ne peut pas contenir d'arc (l, m), $l \in]i, k[$ et $m \in]k, j]$. Les autres arcs de A_{Max} sont donc circonscrits à $]i, k[$(resp. $]k, j]$) et ont déjà été correctement calculées

par $\theta_{(i+1,k-1)}$ (resp. $\theta_{(k+1,j)}$). Il suffit alors d'ajouter 1 (ou une contribution à l'énergie de la structure) à la somme de ces deux nombres pour sanctionner la contribution de l'appariement (i,k) à la structure optimale.

On peut donc calculer les $\theta_{[i,j]}$ à partir des valeurs des $\theta_{[i',j']}$, $[i',j'] \subset [i,j]$ grâce à la récurrence exhibée. ■

Une fois les $\theta_{[i,j]}$ calculés, on peut reconstruire récursivement la structure secondaire solution en repartant de $[1,n]$, puis en testant à chaque étape lequel des termes (a) ou (b_k) l'emporte dans le max de la récurrence. On choisit alors d'apparier ou non la première base, et on relance récursivement le mécanisme de reconstruction sur le ou les intervalles apparaissant dans le terme choisit. Il en résulte l'algorithme suivant :

Algorithme 3 Algorithme MAX-ARCS-WATERMAN : Adaptation de Nussinov/Waterman

Procédure FillMatrix(θ,n,A) :
$\theta_{\varnothing} \leftarrow 0$
pour tout $m \in [0, n-1]$ **faire**
  pour tout $i \in [1, n-m]$ **faire**
    $j \leftarrow i+m$
    $\theta_{[i,j]} \leftarrow \theta_{[i+1,j]}$
    pour tout $(i,\alpha) \in A,\ \alpha \leq j$ **faire**
      $\theta_{[i,j]} \leftarrow \max(1+\theta_{[i+1,\alpha-1]}+\theta_{[\alpha+1,j]}, \theta_{[i,j]})$
    fin pour
  fin pour
fin pour

Fonction Traceback(θ,i,j,n) :
si $i \geq j$ **alors**
  return $\varnothing$
fin si
si $\theta_{[i,j]} = \theta_{[i+1,j]}$ **alors**
  return Traceback$(\theta, i+1, j, n)$
fin si
pour tout $(i,\alpha) \in A,\ \alpha \leq j$ **faire**
  si $\theta_{[i,j]} = 1+\theta_{[i+1,\alpha-1]}+\theta_{[\alpha+1,j]}$ **alors**
    return $\{(i,\alpha)\} \cup$ Traceback$(\theta, i+1, \alpha-1, n) \cup$ Traceback$(\theta, \alpha+1, j, n)$
  fin si
fin pour

Quand le nombre de noeud n est supérieur au nombre d'arc m, on pourra, dans une traversée préliminaire de l'instance en temps $\mathcal{O}(n)$, sélectionner les noeuds (en nombre $\mathcal{O}(m)$) porteurs d'au moins un arc, et exécuter l'algorithme sur cette nouvelle instance.

8.3.1 Complexité et problème MAX-INDEPENDANT-SET

Etude de la complexité

On rappelle que l'algorithme de Nussinov, à l'origine de cet algorithme, est en $\mathcal{O}(n^3)$. Cependant, dans l'application de l'algorithme initial à des instances du problème du repliement, chacune des bases était susceptible de former des liaisons avec de l'ordre de $\mathcal{O}(n)$ autres bases, ce qui provoquait l'apparition d'un surcoût linéaire lors de l'exécution des lignes 7 à 9. Dans notre cas, on peut raffiner ce calcul en pratiquant une analyse *amortie* de la complexité.

Théorème 13 :
La complexité de l'algorithme MAX-ARCS-WATERMAN est en $\mathcal{O}(\min(m,n)m)$*, avec n la longueur de la séquence d'ARN considérée et* $m = |A|$ *le nombre d'appariements.*

PREUVE : Intéressons-nous au nombre de la comparaisons de la ligne 8 de l'algorithme. Un appariement (α_i, β_i) est extrait de A (ligne 7) à chaque fois qu'un intervalle $[\alpha_i, \beta_i + k]$ est observé. Donc la ligne 8 est exécutée $n - \beta_i + 1$ fois pour chaque appariement (α_i, β_i). Le nombre $C_8(n, A)$ de passage par la ligne 8 est donc donné par la formule suivante :

$$C_8(n,A) = \sum_{(\alpha_i,\beta_i)\in A} n - \beta_i + 1 \leq (n+1)m - m = nm$$

De plus, on doit évaluer séparément le nombre d'exécutions de la ligne 7 afin de se faire un idée plus complète de la complexité. En effet, on pourrait imaginer de nombreuses exécutions de la ligne 7 pour lesquelles aucun candidat n'existerait dans A.

$$C_7(n) = \sum_{i=0}^{n-1} n - i = \frac{n(n+1)}{2}$$

La complexité *totale* de la phase de remplissage de la matrice de programmation dynamique est donc en $\mathcal{O}(n^2 + nm)$. La remontée (fonction `Traceback`) est linéaire. Quand $n > m$, il n'existe qu'au plus m noeuds porteurs d'un arc. La passe préliminaire sélectionne au plus $\mathcal{O}(m)$ noeuds avant d'executer l'algorithme. La contribution de n à la complexité ne peut donc pas dépasser m, d'où la complexité en $\mathcal{O}(\min(m,n)^2 + \min(m,n)m) \equiv \mathcal{O}(\min(m,n)m)$ annoncée. ■

8.3.2 MAX-INDEPENDANT-SET pour les graphes circulaires

Comme on l'a vu, le problème PLANAR-MAX-ARCS inclut le problème du repliement dans le modèle de Nussinov. Il est aussi relié étroitement au problème du MAX-INDEPENDANT-SET (MIS) pour les graphes circulaires.

Définition 24 (MAX-INDEPENDANT-SET) :
Etant donnée $G = (V, A)$ *un graphe, renvoyer* $V' \subset V$*, tel que*

$$\forall a, b \in V', (a,b) \notin A.$$

Dans le cas général, ce problème est NP-Complet. Cependant, appliqué à des familles de graphes présentant des propriétés additionnelles, il est parfois résoluble en temps polynomial. C'est le cas pour les graphes circulaires :

Définition 25 (Graphe circulaire) :
Un graphe circulaire est un graphe isomorphe au graphe d'intersection d'un ensemble de cordes sur un cercle. □

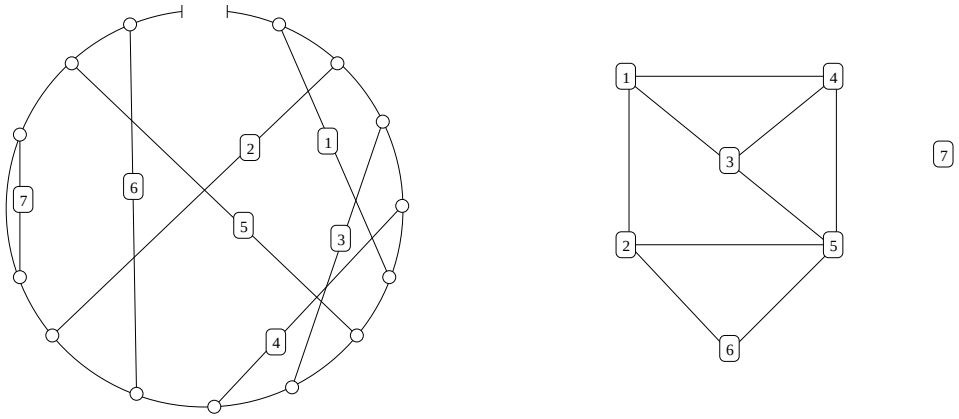


Fig. 8.4: Ensemble de cordes et graphe circulaire associé. Une des solution au problème MIS est l'ensemble des noeuds $\{1,6,7\}$

On reconnaît dans l'ensemble de cordes de la figure 8.4 une structure très similaire à nos objets de prédilection. En particulier, il est possible de transformer une structure générale en un ensemble de cordes d'une telle façon que la solution au problème MIS dans ce graphe puisse être adaptée en une solution pour PLANAR-MAX-ARCS pour l'instance d'origine. Pour cela, il suffit de *rendre incompatibles* tous les arcs ayant un noeud en commun, comme l'illustre la figure 8.5.

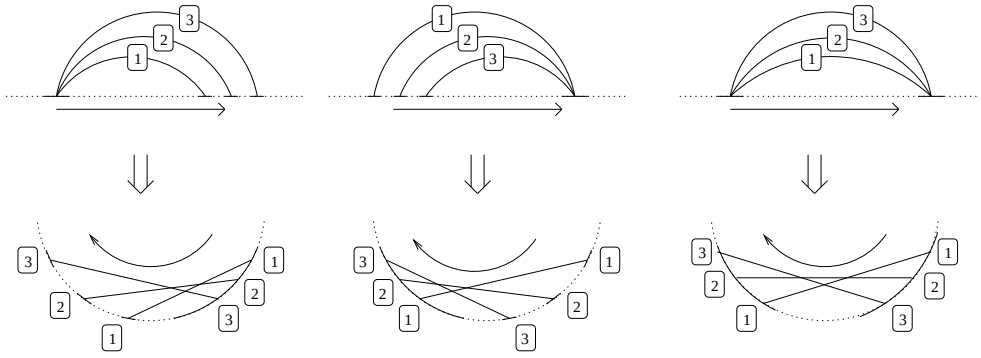


Fig. 8.5: Transformation d'une instance de structure d'ARN en représentation *ensemble de cordes* d'un graphe circulaire.

Il existe une littérature riche s'attachant à résoudre ce problème en la plus faible complexité polynomiale possible. En particulier, Supowit résout ce problème dans [95] par une approche *programmation dynamique* similaire à celle présentée ici, obtenant un algorithme en $\mathcal{O}(N^2)$, où N est le nombre de noeuds du graphe circulaire. Cependant, la nécessaire transformation de l'instance d'origine en un *ensemble de cordes* provoque l'apparition de $\mathcal{O}(m)$ noeuds supplémentaires dans le graphe circulaire associé. L'utilisation d'un tel algorithme pour résoudre notre problème entraîne une complexité en $\mathcal{O}(m^2)$, ne tirant pas profit des cas, nombreux dans l'ARN, où les noeuds sont de forts degrés.

Une autre solution, proposée par Apostolico et al. dans [1], et qui est à l'heure actuelle tenue comme la meilleure pour ce problème, revendique une complexité en $\mathcal{O}(Nd)$, où N est le nombre de noeuds et d le nombre maximal de cordes *survolant* un point du cercle dans la représentation *ensemble de cordes* du graphe circulaire. Encore une fois, quand $m \gg n$, c'est à dire dans les cas où l'ARN d'origine est très dense, et en remarquant que $d \in \mathcal{O}(m)$, borne atteinte par une clique de taille m, la complexité de l'algorithme revient à $\mathcal{O}(m^2)$, quand celle de notre algorithme reste en $\mathcal{O}(nm)$. Cependant, il est fort probable que l'algorithme d'Apostolico soit supérieur en moyenne au notre, dans la mesure où le paramètre d peut aussi être interprété comme la hauteur maximale de la marche aléatoire, dite de Dyck, associée à l'instance du problème (Ensemble de cordes $\Leftrightarrow$ Mot bien parenthèsés). Ce paramètre est connu pour se comporter comme $\mathcal{O}(\sqrt{m})$ en moyenne. Le cas au pire nous reste cependant favorable.

8.4 Application

Cet algorithme a été implémenté, et appliqué à toutes les structures secondaires de la base de donnée CRW [22], où les structures sont déterminées par une méthode comparative, c'est à dire n'excluant *a priori* pas les interactions *tertiaires*. On trouvera dans la figure 8.6 un exemple du type de structure obtenue.

Dans les structures issues de cette base de données, la proportion totale des appariements impliqués dans des interactions tertiaires est de l'ordre de 2%.

Il appartient aux perspectives de ces travaux d'appliquer cet algorithme aux structures issues de la NDB, ramenées à des structures grâce à MC-Sym ou RnaView.

8.5 Discussion et perspectives

On a présenté un algorithme issu de la programmation dynamique pour la planarisation de structure *générale* d'ARN maximisant le nombre d'arcs conservés. La complexité de cet algorithme est en $\mathcal{O}(\min(m,n)m)$ en temps et $\mathcal{O}(\min(m,n)^2 + n)$ en espace. Cet algorithme est, à notre connaissance, le premier algorithme non-heuristique pour la planarisation de structure d'ARN. En outre, il possède une complexité concurrentielle avec les meilleurs algorithmes disponibles pour le problème MIS dans les graphes circulaires. Enfin, il se transpose naturellement dans un domaine pondéré par une fonction σ, c'est à dire dans des modèles très simples d'énergie libre, comme celui de Nussinov, où les différentes paires de bases se voient associer

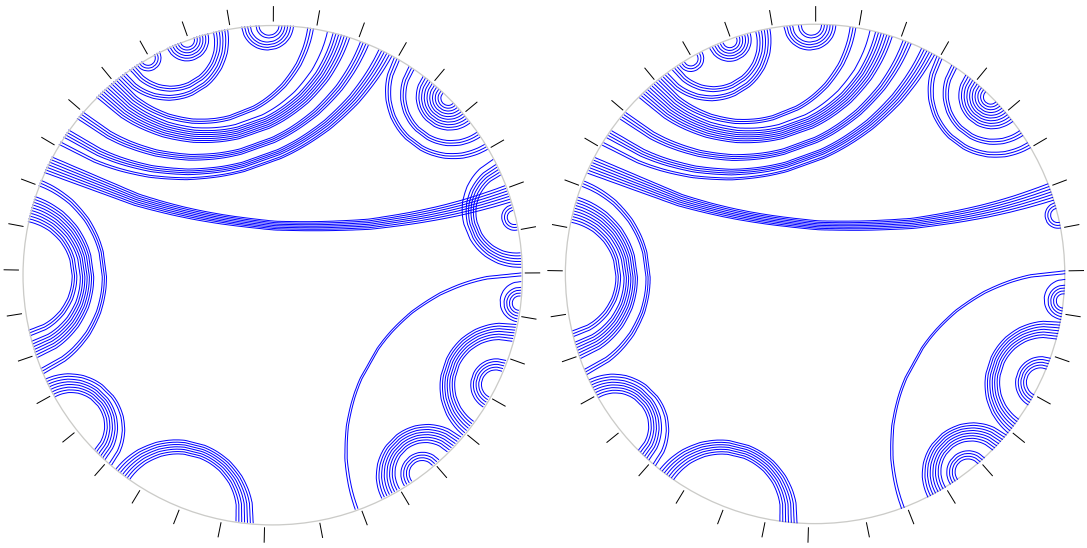


Fig. 8.6: Version originale et aplanie de la structure d'un intron du groupe 1 chez *B. atropupurea*

des poids différents[2], c'est à dire des contributions différentes à l'énergie libre de la structure obtenue.

Un algorithme similaire peut être construit pour des modèles thermodynamiques plus complexes, en s'inspirant des algorithmes de programmation dynamique proposés en réponse au problème du repliement de l'ARN. Une étude spécifique dans le cas du modèle de Turner sous-tendant l'algorithme MFold permettrait de conclure sur la possibilité d'optimiser cet algorithme dans le cas de la planarisation, comme nous avons pu le faire dans le modèle de Nussinov-Jacobson.

[2] Typiquement, on utilise $\sigma(\{G,C\}) = 3$, $\sigma(\{A,U\}) = 2$ et $\sigma(\{G,U\}) = 1$.

Partie III

DÉNOMBREMENT ET GÉNÉRATION ALÉATOIRE DES CHEMINS CULMINANTS

Plan et contributions

Les études présentées dans ce dernier chapitre sont initialement motivées par l'analyse d'un algorithme heuristique de recherche de similarités, dont la sensibilité s'avère étroitement liée à la probabilité de présence d'un motif dans le cheminement de l algorithme.

On s'est dans dans un premier temps intéressé à la génération aléatoire de ces séquences, qu'on a baptisé *chemins culminants* en référence aux cheminements du score lors d'une lecture *de gauche à droite*. De tels chemins doivent être positifs et culminer, c'est à dire terminer à la plus grande hauteur jamais atteinte, sous peine de contenir une partie de plus grande amplitude. Une étude préliminaire révèle que ces chemins sont codés par un langage rationnel quand la hauteur finale est bornée préliminairement, et contextuel sinon.

Une adaptation de la méthode récursive présentée précédemment permet alors de trouver un algorithme de complexité polynomiale pour la génération aléatoire de ces chemins. De plus, une transposition de la phase de dénombrement préliminaire en un algorithme de programmation dynamique donne une évaluation exacte de la sensibilité d'une graine dans ce modèle[53]. L'application de cet algorithme met en évidence un biais dans l'évaluation de la sensibilité par des méthodes ne tenant pas compte de l'aspect culminant. Cependant, sa complexité, bien que polynomiale, reste élevée et rendrait impossible l'évaluation de la sensibilité dans des séquences de tailles de l'ordre du millier de base.

On poursuit alors cette étude de ces chemins sur un plan plus combinatoire, afin de trouver une description de ces chemins permettant une génération aléatoire plus efficace, ou bien des propriétés énumératives permettant une application de la génération aléatoire par rejet. Celle ci consiste à engendrer itérativement des objets dans un surensemble de l'ensemble étudié jusqu'à obtenir un objet de l'ensemble souhaité. Si le surensemble est sympathique, c'est à dire qu'il admet une décomposition rendant efficace la génération aléatoire, et que les cardinalités des deux ensembles ne sont pas trop déséquilibrées, alors on tient un algorithme de génération aléatoire pour nos objets. Dans le cas des chemins culminants, L. Noé et G. Kucherov avaient déjà tenté d'engendrer des chemins culminants par rejet à partir des séquences de $(m, \overline{m})^*$ prises uniformément, et avaient constaté des complexités respectivement linéaires et exponentielles en fonction valeurs relatives des bonus/malus associés aux conservations/altérations. Nous avons donc étudié avec M. Bousquet-Mélou les propriétés énumératives de ces chemins, qui ont corroboré les observations et intuitions formulées par mes précédents coauteurs. Les comportements énumératifs ont permis la formulation d'algorithmes de génération aléatoire par rejet linéaires en nombre d'opération arithmétiques.

Enfin, l'aspect contextuel du langage associé aux chemins culminants, n'ayant pas empêché la mise en oeuvre d'une légère variante de la méthode récursive, suggère la possibilité d'étendre ce paradigme. Nous proposons donc un cadre unificateur pour décrire et analyser l'approche récursive de génération aléatoire initiée par Wilf et étendue par Flajolet et al aux structures décomposables, en l'étendant à certaines classes descriptibles itérativement.

Contributions :

- Propriétés linguistiques des langages associés aux chemins culminants
  Section 9.3, page 127

- Algorithme de génération aléatoire basé sur une approche récursive pour les chemins culminants
  Section 10.1, page 130

- Equivalent de l'algorithme de Keich pour les chemins culminants, mise en évidence d'un biais dans l'estimation de la sensibilité
  Section 10.2.2, page 132

- Par la méthode du noyau, obtention de la série génératrice des chemins culminants et asymptotique en $2^n/4n$ du nombre de chemins culminants à pas $+1,-1$ de taille n
  Section 10.2.2, page 132

- Cas $+1,-1$, génération linéaire en moyenne grâce au langage miroir des préfixes de Dyck généralisés
  Section 11.3, page 135

- Algorithmes de rejet dans les cas de dérive positive ou nulle, linéaires et sans précalcul (Rejet anticipé)
  Section 11.3, page 135

- Un cadre unificateur pour la méthode récursive de génération aléatoire
  Section A, page 159

9. INTRODUCTION

La génération aléatoire de séquences et structures génomiques permet aussi l'évaluation de la sensibilité des algorithmes heuristiques. De tels algorithmes, qui gagnent en complexité ce qu'ils perdent en capacité prédictive, permettent le traitement de grands volumes de données en des temps raisonnables, ce qui constitue un atout de taille dans l'optique d'une application à la bioinformatique. De plus, leurs bases théoriques, c'est à dire les problèmes informatiques qu'ils sont censés résoudre, reflètent l'état des connaissances biologiques sur les données manipulées, c'est pourquoi un déterminisme coûteux en temps de calcul, voire rédhibitoire, n'est pas essentiel. Cependant, il peut être intéressant, afin d'établir des intervalles de confiance par exemple, ou bien juste pour augmenter l'efficacité de la méthode à temps de calcul constant, de disposer de mécanismes d'évaluation de la sensibilité des algorithmes.

Dans ce chapitre, nous aborderons l'étude de la sensibilité des algorithmes heuristiques de recherche de similarité entre deux séquences [4; 71; 21]. Au cours de sa thèse, L. Noé s'intéresse à la conception de graines, un paramètre de certains algorithmes caractérisant des masques pour des motifs parfaitement conservés entre les séquences à partir desquelles l'algorithme va construire des zones candidates. Ces zones sont ensuite étendues jusqu'à ce qu'une certaine condition de seuil ne soit plus respectée, et une région d'amplitude maximale est alors extraite de la zone étendue. Cette amplitude est relative à une fonction de score additive calculée séquentiellement, qui pénalise deux bases différentes et favorise deux bases identiques. On voit donc apparaître deux origines à la non-exhaustivité dans l'exploration par l'algorithme de l'espace de recherche : La recherche de régions conservées pour l'ancrage initial et la condition de seuil lors de l'extension. La question est alors de savoir quelle est la proportion des similarités qui sont effectivement détectables par l'algorithme malgré l'heuristique d'ancrage initial. De nombreux auteurs se consacrent à ce problème, en le ramenant au calcul de la probabilité d'observer la graine dans une séquence aléatoire, constituée de conservation/altération, et engendrée selon un modèle uniforme ou markovien. Ils négligent en cela implicitement la sélection d'une région d'amplitude maximale. Or cette condition a un impact important sur les séquences considérées, et sa non-considération résulte en l'apparition d'un biais dans les sensibilités calculées, comme nous allons le mettre en évidence (Travaux publiés dans [53]).

9.1 Recherche de similarité : Principe des algorithmes heuristiques

Historiquement, le premier algorithme pour la recherche *locale* de similarité est le classique Smith-Waterman [90]. En utilisant une approche *programmation dynamique*, il extrait la meilleure similarité entre deux séquences S_1 et S_2. Le critère maximisé est lié à un système de score assez

simple, et sa complexité est en $\Theta(n_1 n_2)$, où n_1 et n_2 sont les tailles des deux séquences en jeu. Avec la multiplication des données disponibles et surtout de leur taille, une telle complexité est vite devenue rédhibitoire, et des solutions heuristiques ont vu le jour. Parmi celles-ci, BLAST [4], qui *optimise* cette complexité quadratique de la façon suivante :

1. **Décomposition :** Une des deux séquences, par exemple S_2, est *cassée* en motifs de taille k. On alors construit une liste $\mathscr{L}$ de motifs *ressemblant* à un motif de la liste, c'est à dire ayant un score de similarité supérieur à un certain seuil.

2. **Recherche :** On recherche alors des occurrences *exactes* des motifs de $\mathscr{L}$ dans S_1. On obtient alors un ensemble $\mathscr{P}$ de positions candidates, appelées *ancrages*.

3. **Extension :** Pour chacun des ancrages, on part du score de similarité, et on *étend* l'alignement dans les deux directions, jusqu'à ce que la fin de la séquence soit rencontrée, ou qu'un certain seuil inférieur X pour le score soit dépassé (heuristique **XDrop**). On extrait alors de l'alignement obtenu le sous-alignement de score maximal, et on stocke l'alignement candidat.

Cependant, on a pu déplorer, dans la communauté, un manque de sensibilité de la part de cette approche, particulièrement dans des génomes très variables. Ce défaut était attribué à la nécessité, au cours de l'étape de *recherche*, de trouver des occurrences *exactes* des motifs de $\mathscr{L}$. C'est pourquoi on retrouve, dans [21] par exemple, un affaiblissement de la condition d'ancrage au cours de la phase de *recherche*. L'idée est alors d'ancrer en des positions faisant apparaître un occurrence *approchée* d'un motif de $\mathscr{L}$. On utilise pour cela un masque, ou *graine*, une séquence composée de 0 et de 1 qui autorise explicitement (0) des variations du motif-cible en certaines positions. C'est à l'influence de cette graine sur la sensibilité de l'algorithme que nous nous intéressons dans la suite de ce chapitre.

9.2 Formalisation du problème

Nous assimilerons dans la suite de notre étude le cheminement du score dans une lecture *de gauche à droite* d'une homologie sélectionnée à une marche aléatoire uniforme, composée de pas $m = (+1, +a)$ et $\overline{m} = (+1, -b)$. Ces deux types de pas correspondent respectivement à une conservation (Codée par m) et à une altération (Codée par $\overline{m}$) dans l'alignement modélisé par la marche. Les pas m correspondant à une conservation se verront associer un bonus $+a$, et les pas $\overline{m}$ un malus $-b$. La prise en compte de probabilités différentes pour les différents pas, ainsi que de différents types de pas pour différentes altérations, feront l'objet d'une discussion ultérieure. Ces marches aléatoires doivent alors posséder deux propriétés supplémentaires pour prétendre à modéliser le cheminement d'un score d'alignement :

- **Culminance** : Si la hauteur atteinte à la fin de la marche n'est pas la plus élevée jamais atteinte pendant son parcours, alors il existe un préfixe de la marche qui optimise la hauteur finale. La marche ne représente alors pas un alignement valide, car celui-ci se doit d'être de score maximal par construction.

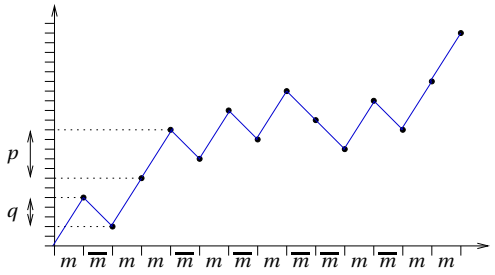


Fig. 9.1: Encodage d'un chemin par un mot sur $\{m, \overline{m}\}^*$

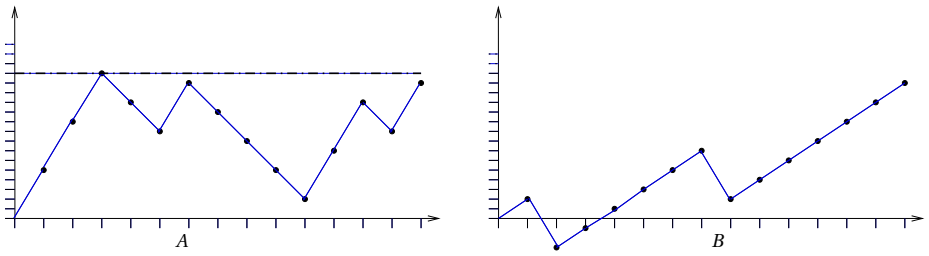


Fig. 9.2: Deux marches aléatoires non-culminantes, car violant respectivement les conditions de positivité (**A**) et de culminance (**B**)

- **Positivité** : Si la marche atteint des altitudes négatives, alors il est possible de trouver un suffixe de cette marche de score supérieur, en partant de n'importe quel point négatif. Le score de ce l'alignement correspondant à cette nouvelle marche est alors supérieur à celui de l'alignement original, ce qui est impossible par construction de l'alignement.

Définition 26 (Chemins culminants) :
Soit $n \in \mathbb{N}$, un chemin culminant est une marche $(0,\eta_0),\ldots,(n,\eta_n)$ sur $\mathbb{Z}^2$ commençant en $(0,0)$, composée de pas $\{(1,a),(1,-b)\}$ et satisfaisant les deux propriétés suivantes :

$$\forall i \in [0,n-1], \quad \eta_i < \eta_n \quad \textit{(Culminance).}$$

$$\forall i \in [1,n], \quad \eta_i > 0 \quad \textit{(Positivité)}$$

Puisque nous allons nous intéresser à la génération aléatoire de ces marches, nous allons manipuler le langage des codages associés à des chemins culminants. Pour cela, nous utiliserons un codage naturel, qui consiste à remplacer chaque occurrence de $(1,a)$ par m et chaque occurrence de $(1,b)$ par $\overline{m}$ dans la séquences des pas composants la marche.
Soit $\phi_{a,b} : \{\overline{m}, m\}^* \to \mathbb{N}$ une fonction de hauteur qui associe à chaque codage d'une marche ω sa hauteur finale telle que

$$\phi_{a,b}(\omega) = a|\omega|_m - b|\omega|_{\overline{m}}$$

Les contraintes sur la marche se transposent alors naturellement en contraintes sur leur codage.

Définition 27 (Langage des mots culminants) :
Le langage des mots culminants est l'ensemble $\mathscr{C}^{a,b} \in \{\overline{m}, m\}^*$ *de mots* ω *tels que, pour tout préfixe non-vide* ω' *de* ω *:*

$$\phi_{a,b}(\omega') > 0 \quad \textit{(Positivité)},$$

et, pour tout préfixe propre ω' *de* ω *:*

$$\phi_{a,b}(\omega') < \phi_{a,b}(\omega) \quad \textit{(Culminance)}.$$

On rappelle que le préfixe ω' d'un mot ω est *propre* si $\omega' \neq \omega$. Nous noterons $\mathscr{C}^{a,b\Rightarrow k}$ le sous-ensemble des mots de $\mathscr{C}^{a,b}$ contenant les mots terminant à hauteur k. De plus, nous restreindrons notre étude aux cas où a et b sont des entiers premiers entre eux, cas auquel on peut toujours se ramener en remarquant que les culminants sont stables par homothétie($\Leftrightarrow \mathscr{C}^{a,b} = \mathscr{C}^{\alpha a,\alpha b}$). Enfin, on utilisera le seul terme chemin culminant pour faire référence, selon la situation, à un chemin culminant ou à son codage.

9.3 Propriétés du langage associé

Quand on s'intéresse à la génération aléatoire de mots d'un langage, une question naturelle est : *A quelle catégorie des langages formels mon langage appartient-il ?* En effet, si la réponse place le langage étudié dans la catégorie des langages rationnels ou algébrique, alors la découverte d'un automate ou d'une grammaire décrivant le langage permet la dérivation automatique d'algorithmes de génération aléatoire uniforme efficaces [26; 42; 76]. Nous allons donc étudier les propriétés du langage des chemins culminants.

9.3.1 Chemins culminants de hauteur bornée

Théorème 14 :
Pour tout $a, b \in \mathbb{N}^*$ *et* $k \in \mathbb{N}$*, le langage* $\mathscr{C}^{a,b\Rightarrow k}$ *des chemins culminants atteignant une hauteur finale* k *est rationnel.*

PREUVE : Quand la hauteur finale k est imposée a priori, les chemins culminants circulent dans une bande de hauteur k, c'est à dire dans un espace fini de positions atteignables. De plus, les *futurs* autorisés à partir d'une position dépendent uniquement de la position en cours et de la valeur de k. Il est donc possible de construire un automate fini qui reconnaît exactement les chemins de $\mathscr{C}^{a,b\Rightarrow k}$.

Les états de cet automate sont les auteurs accessibles $[0, k]$, plus un état *poubelle* $\perp$. 0(resp. k) est alors l'état initial(resp. final) de l'automate, et la fonction de transition δ est telle que :

$$\begin{array}{rclcrcl}
\delta(q,m) & = & \left\{\begin{array}{ll} q+a & \text{if } q \leq k-a, \\ \perp & \text{otherwise} \end{array}\right. & & \delta(q,\overline{m}) & = & \left\{\begin{array}{ll} q-b & \text{if } q > b, \\ \perp & \text{otherwise} \end{array}\right. \\
\delta(k,_) & = & \perp & & \delta(\perp,_) & = & \perp .
\end{array}$$

Il est clair que cet automate rejette tout mot cherchant à descendre en dessous de 0(resp. au dessus de k) vers l'état *poubelle* $\perp$, où il restera jusqu'à la fin de sa lecture et sera rejeté. De plus, il

accepte uniquement les mots dont la lecture termine en l'état k, jamais atteint précédemment. Il reconnaît donc exactement $\mathscr{C}^{a,b\Rightarrow k}$. Comme l'espace d'états est fini, alors $\mathscr{C}^{a,b\Rightarrow k}$ est un langage régulier. ■

Cet automate peut alors être utilisé pour dériver un algorithme de génération aléatoire uniforme pour $\mathscr{C}^{a,b\Rightarrow k}$ et linéaire [47; 45] sur la taille des objets engendrés, par une application de la méthode récursive similaire à celle présentée dans le cas des mots du langage d'une grammaire non-contextuelle. On précalculera, dans un premier temps, pour tout état q et tout $i \leq n$, les nombres N_q^i de mots de taille i reconnus par l'automate en partant de l'état q. Une fois ces nombres calculés, un chemin culminant de taille n et de hauteur finale k peut être engendré en *marchant* dans l'automate, en partant de 0 et en prolongeant à chaque étape un chemin arrivant en q par un pas montant m vers un état $q+a$ (resp. descendant $\overline{m}$ vers $q-b$) avec probabilité $N_{q+a}^{n-i-1}/N_q^{n-i}$ (resp.$N_{q-b}^{n-i-1}/N_q^{n-i}$). On obtient alors un algorithme en complexité arithmétique linéaire.

9.3.2 Chemins culminants : Le cas général

En général, quand on veut prouver qu'un langage n'est reconnu par aucune grammaire non-contextuelle, on utilise deux outils familiers des étudiants en Licence d'Informatique : La version algébrique du lemme de l'étoile, ou le lemme d'Ogden. Ici, une propriété amusante du langage des chemins culminants est qu'il n'est mis en défaut par aucun de ces deux lemmes, bien qu'il s'avère au final appartenir à une classe supérieure à celle des langages non-contextuel, comme l'établit la proposition suivante.

Proposition 2 :
Pour tout $a, b \in \mathbb{N}$, le langage $\mathscr{C}^{a,b}$ des chemins culminants n'est pas non-contextuel.

PREUVE : On rappelle que l'intersection d'un langage non-contextuel et d'un langage rationnel est non-contextuel [48]. De plus, d'après la version algébrique du lemme de l'étoile [48, Theorem 4.7], pour tout langage $\mathscr{L}$, il existe $n \in \mathbb{N}$ tel que tout mot $\omega \in \mathscr{L}$ de taille au moins n admet une factorisation $\omega = u.x.v.y.t$ satisfaisant les propriétés suivantes :

1. $|x.y| \geq 1$,
2. $|x.v.y| \leq n$,
3. $\forall \ell \geq 0, \omega_\ell := u.x^\ell.v.y^\ell.t \in \mathscr{K}$.

Soit $\mathscr{R}$ le langage rationnel défini par l'expression rationnelle

$$\mathscr{R} = \mathscr{L}(m^*.\overline{m}^+.m^*)$$

et qu'on peut voir comme le langage des chemins en *zig-zag*. Soit $\mathscr{K} = \mathscr{C}^{a,b} \cap L$. On prouve facilement que

$$\mathscr{K} = \{m^i.\overline{m}^j.m^k \mid i > 0,\ bj < ai \text{ et } bj < ak\}.$$

On va raisonner par l'absurde, en supposant avéré $\mathscr{C}^{a,b}$ non-contextuel. Si $\mathscr{C}^{a,b}$ est non-contextuel, alors $\mathscr{K}$ l'est aussi.

Comme a et b sont premiers entre eux, alors il existe $i > n$ et $j > n$ tels que $ia - jb = 1$ (Théorème de Bachet-Bezout). Alors le mot $\omega = m^i \overline{m}^j m^i$ appartient à $\mathscr{K}$. Dans le reste de la preuve, nous appellerons A la première série de pas montants de ω, B la série de pas descendants et C la série finale de pas montants.
Nous résumons dans la table 9.1 pour chacune des décompositions admissibles de ω, une valeur de ℓ pour laquelle le mot ω_ℓ n'appartient pas à $\mathscr{K}$. La raison pour laquelle ω_ℓ n'appartient pas à $\mathscr{K}$ est explicitée dans la colonne de droite.

Position de $x.v.y$	ℓ	ω_ℓ	Condition mise en défaut
A	0	$m^{i-h}.\overline{m}^j.m^i$	Pos. : $\phi(m^{i-h}.\overline{m}^j) = 1 - ah \leq 0$
B	2	$m^i.\overline{m}^{j+h}.m^i$	Pos. : $\phi(m^i.\overline{m}^{j+h}) = 1 - bh \leq 0$
C	0	$m^i.\overline{m}^j.m^{i-h}$	Culm. : $\phi(\omega_\ell) = \phi(m^i) - ah \leq \phi(m^i)$
$A \cup B$			
$\lvert x\rvert_{\overline{m}}\lvert x\rvert_m + \lvert y\rvert_{\overline{m}}\lvert y\rvert_m \neq 0$	2	$m^i.\overline{m}^{h'}.m^h.\overline{m}^j.m^i$	$\omega_\ell \notin \mathscr{K}$ (Trop de pics)
$x = m^h,\ y = \overline{m}^{h'}$			
$\Delta < 0$	2	$m^{i+h}.\overline{m}^{h'+j}.m^i$	Pos. : $\phi(m^{i+h}.\overline{m}^{j+h'}) = 1 + \Delta < 0$
$\Delta = 0$	2	$m^{i+h}.\overline{m}^{h'+j}.m^i$	Culm. : $\phi(m^{i+h}) = a(i+h) \geq \phi(\omega_\ell)$
$\Delta > 0$	0	$m^{i-h}.\overline{m}^{j-h'}.m^i$	Pos. : $m^{i-h}.\overline{m}^{j-h'} = 1 - \Delta \leq 0$
$B \cup C$			
$\lvert x\rvert_{\overline{m}}\lvert x\rvert_m + \lvert y\rvert_{\overline{m}}\lvert y\rvert_m \neq 0$	2	$m^i.\overline{m}^j.m^{h'}.\overline{m}^h.m^i$	$\omega_\ell \notin \mathscr{K}$ (Trop de vallées)
$x = \overline{m}^{h'},\ y = m^h$			
$\Delta \leq 0$	2	$m^i.\overline{m}^{h'+j}.m^{i+h}$	Pos. : $\phi(m^i.\overline{m}^{h'+j}) = 1 - bh' \leq 0$
$\Delta > 0$	0	$m^i.\overline{m}^{j-h'}.m^{i-h}$	Culm. : $\phi(\omega_\ell) = \phi(\omega) - \Delta \leq \phi(m^i)$

Tab. 9.1: Décompositions de $\omega = m^i \overline{m}^j m^i$ et mises en défaut du lemme de l'étoile. i et j sont les valeurs pour lesquelles $ia - jb = 1$ et $\Delta := ah - bh'$.

Dans les cas $A \cup B$ et $B \cup C$, il faut traiter séparément les cas où x et y sont monotones. On remarquera aussi que, du fait de la condition 2 du lemme de l'étoile, le facteur $x.v.y$ ne peut empiéter simultanément sur les régions A et C, ce qui limite le nombre de décompositions à examiner. Comme nous avons décrit toutes les décompositions possibles pour ω, et qu'aucune ne satisfait aux conditions du lemme de l'étoile, alors le langage $\mathscr{K}$ n'est pas non-contextuel, donc $\mathscr{C}^{a,b}$ ne l'est pas non plus. ■

Il n'existe donc pas de grammaire non-contextuelle qui décrive les langages $\mathscr{C}^{a,b}$, la génération aléatoire de mots de ce langage devra donc utiliser d'autres techniques que celles disponibles pour les langages algébriques. Parmi les techniques alternatives à la génération aléatoire de mots d'un langage non-contextuel, l'application de la méthode récursive de Wilf [103], qui transpose une relation de récurrence entre des cardinalités du langage en principe de génération aléatoire uniforme.

10.1 Génération aléatoire : L'approche récursive

C'est sur cette approche que repose l'algorithme de génération aléatoire proposé en [53], linéaire en temps après un précalcul non-linéaire. Les trois idées sous-jacentes à l'algorithme sont résumées ici :

1. Soit $\mathscr{W}$ un langage et $\mathscr{W}_p$ le langage des préfixes de mots de $\mathscr{W}$. Supposons que pour tout $\omega \in \mathscr{W}_p$ et tout $n \geq |\omega|$, nous connaissons le nombre de mots de $\mathscr{W}$ de taille n commençant par ω (Nous appellerons ces mots des *extensions* de ω). Alors il est possible d'engendrer uniformément des mots de $\mathscr{W}$ de taille n en adoptant une approche *pas à pas*.

2. Quand $\mathscr{W} = \mathscr{C}^{a,b}$, le nombre d'extensions de taille n d'un préfixe $\omega \in \mathscr{W}_p$ dépend uniquement de trois paramètres :

   - Le nombre $i = n - |\omega|$ de pas restants,
   - La hauteur finale $j = \phi(\omega)$,
   - La plus grande hauteur h atteinte par ω.

   Soit $c_{i,j,h}$ le nombre d'extensions de taille n pour un préfixe ω.

3. Les nombres $c_{i,j,h}$ obéissent à la récurrence suivante :

$$\begin{aligned} c_{i,j,h} &= c_{i-1,j+a,\max(h,j+a)} + \mathbb{1}_{j>b}\, c_{i-1,j-b,h} \quad \text{pour } i > 1, \\ c_{1,j,h} &= \mathbb{1}_{j+a>h}. \end{aligned}$$

Comme les trois paramètres i, j et h sont bornés par n, an et an respectivement, le précalcul des nombres $c(i,j,h)$ requiert $\Theta(n^3)^*$ opérations arithmétiques (et requiert autant d'espace). Ensuite, la génération proprement dite de mots de taille n requiert $\mathscr{O}(n)^*$ opérations arithmétiques. Cependant, on doit tenir compte du surcoût lié à la manipulation de grands nombres,

c'est à dire de nombres croissant exponentiellement avec n. C'est ici le cas et, les nombres $c_{i,j,h}$ étant exponentiels en i, la complexité *binaire* en temps et en espace associée à cette étape est susceptible de croître jusqu'à $\Theta(n^4)$. En utilisant une technique adaptée de Denise et Zimmermann [30], nous pouvons limiter les complexités binaires des phases de dénombrement et de génération respectivement à $\mathcal{O}(n^{3+\varepsilon})$ et $\mathcal{O}(n^{1+\varepsilon})$.

On peut aussi précalculer les rapports

$$p_m = \frac{c_{i-1,j+a,max(h,j+a)}}{c_{i,j,h}}$$

qui correspondent à la probabilité d'un pas ascendant. Ces nombres sont rationnels, donc admettent un développement binaire périodique. On peut donc noter sous une forme *condensée* ces développements, en les décomposant sous la forme (préfixe,période). En tirant des bits aléatoires correspondant aux termes successifs du développement binaire d'un nombre aléatoire compris dans $[0,1[$, on arrive à choisir un pas m ou $\overline{m}$ en temps constant en moyenne.

10.2 Calcul de sensibilité de la recherche heuristique de similarité

Après avoir modélisé le cheminement du score, on s'intéresse dans un premier temps au calcul de la sensibilité associée à une *graine* [21; 59; 20]. Une *graine* est un mot s composé de 1 et de 0, ou *masque*, dans lequel les 1 correspondent à des conservations obligatoires, et les 0 à une conservation optionnelle (altération autorisée). L'aspect heuristique des algorithmes [21; 59; 20] réside dans le fait qu'ils imposent qu'une homologie de séquence respecte exactement les 1 du masque.

Soit $\mathcal{D}(s) \subset \{m, \overline{m}\}^*$ l'ensemble des séquences décrites par s, telle que :

$$\mathcal{D}(1.s') = \{m\}.\mathcal{D}(s') \quad \mathcal{D}(0.s') = \{m, \overline{m}\}.\mathcal{D}(s') \quad \mathcal{D}(\varepsilon) = \varnothing$$

On dit alors qu'une graine s *reconnaît* le codage ω d'un alignement ssi $\omega = \alpha\omega'\beta$ tel que $\omega' \in \mathcal{D}(s)$. La *sensibilité* associée à une graine est donc égale à la probabilité qu'elle reconnaisse un alignement.

10.2.1 Échantillonnage

Il est possible, pour estimer une probabilité, d'effectuer l'analogue d'un sondage. Il existe alors une relation entre l'incertitude, l'erreur relative et le nombre d'expériences n de tirages, supposées indépendants, à effectuer. En particulier pour une incertitude de 1%, le nombre de séquences à engendrer afin d'obtenir une erreur relative sur l'estimation de 1% est de 22500. Il est à noter que ce nombre est indépendant de la taille des séquences considérées. Dans la mesure où le rapport entre les complexités des phases de précalcul et de génération est au moins en $\mathcal{O}(n^2)$, cette multiplication du nombre de générations ne représente pas réellement un surcoût majeur. De plus, la reconnaissance d'une graine s peut être effectuée *à la volée* au prix d'un facteur $|s|$. Cependant on peut aussi calculer *exactement* la sensibilité d'une graine.

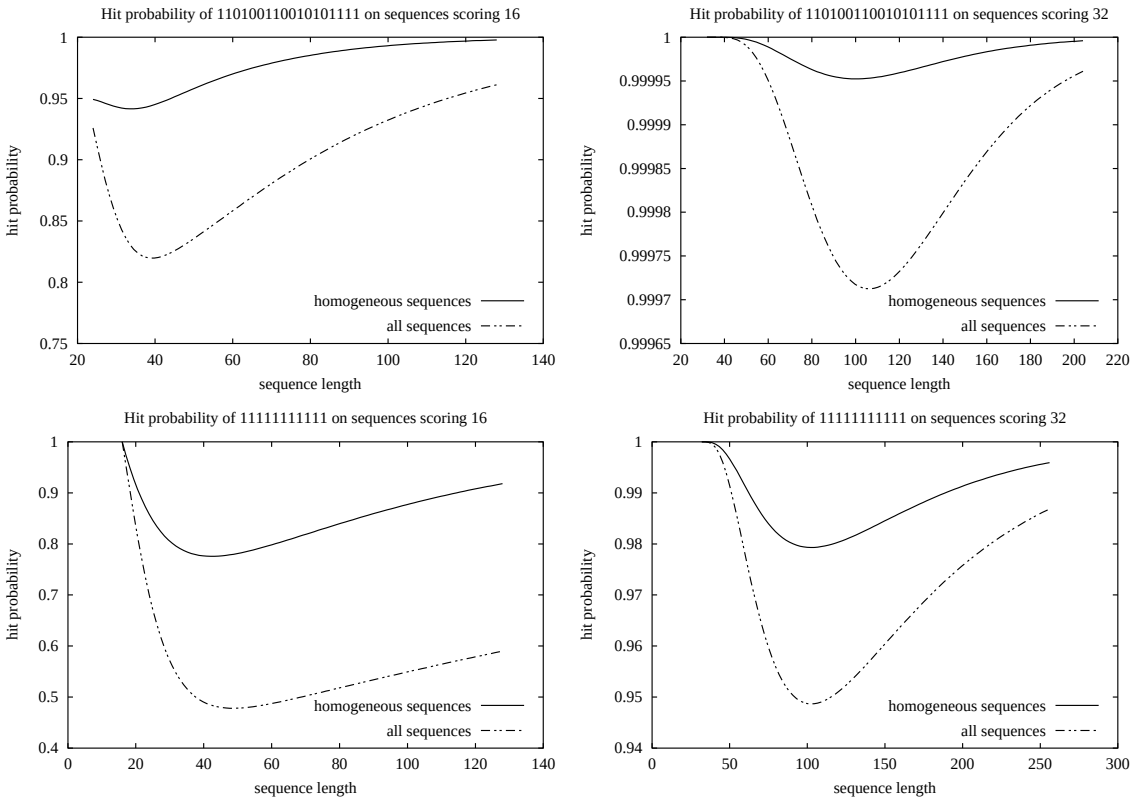


Fig. 10.1: Mise en évidence du biais dans les approches ne tenant compte ni de la culminance ni de la positivité des chemins.

10.2.2 Adaptation de l'algorithme de Keich

Dans [52], Keich et al. proposent un algorithme de programmation dynamique pour le calcul **exact** de la sensibilité d'une graine sous un modèle de Bernoulli pour les séquences de $\{m,\overline{m}\}^n$, n fixé. Nous adaptons cet algorithme en utilisant les quantités $\mathrm{c}_{i,j,h}$ calculées au cours de la phase préliminaire. L'idée générale de cet algorithme consiste en une décomposition des $\mathrm{c}_{i,j,h}$ en $\mathrm{c}^{\omega}_{i,j,h}$ dont la sémantique est *nombre de suffixes ω_s de taille i distincts pour un préfixe ω_p de chemin culminant terminant en j, de hauteur maximale h et terminant par la séquence ω, et tels que $\omega_p\omega_s$ est reconnu par s*. Si ω est reconnu par s, alors $\mathrm{c}^{\omega}_{i,j,h} = \mathrm{c}_{i,j,h}$, c'est à dire que tous les suffixes suivants comptent comme des suffixes de chemins culminants reconnus par s.

$$\mathrm{c}^{\alpha\omega'}_{i,j,h} = \begin{cases} \mathrm{c}_{i,j,h} & \text{Si } s \text{ reconnait } \omega \\ \mathrm{c}^{\omega' m}_{i-1,j+a,\max(h,j+a)} + \mathbb{1}_{j>b}\,\mathrm{c}^{\omega'\overline{m}}_{i-1,j-b,h} & \text{Sinon} \end{cases}, \ \forall i > 1$$

$$\mathrm{c}^{\omega}_{1,j,h} = \begin{cases} \mathrm{c}_{1,j,h} & \text{Si } s \text{ reconnait } \omega \\ 0 & \text{Sinon} \end{cases}$$

Une implémentation naïve de cette récurrence donne une complexité en $\mathcal{O}(|s|2^{|s|}n^3)$, le facteur $|s|$ étant dû au test de reconnaissance par s de ω et $2^{|s|}$ étant la taille de l'espace des ω. Le facteur

$|s|2^{|s|}$ peut être remplacé par la taille, fonction de s, de l'automate fini déterministe minimal reconnaissant le langage $m,\overline{m}^*.\mathscr{D}(s)$. On remplace alors le suffixe ω dans la récurrence ci dessus par l'état actuel dans l'automate, et le test de reconnaissance revient juste à tester la finalité de l'état en cours.

On met alors en évidence dans la figure 10.1 un biais dans l'estimation de la sensibilité par des approches ne tenant pas compte des propriétés de culminance et de positivité. En particulier, la non-prise en compte des deux contraintes (positivité et culminance) revient à sous-estimer (courbes *all sequences*, figure 10.1), parfois grandement, la sensibilité constatée en se limitant aux séquences culminantes(courbes *homogenous sequences*).

Cependant, cette étude initiale ne s'était pas révélée pleinement satisfaisante. En effet, on n'avait pas encore bien analysé et compris la structure combinatoire de ces objets. Leur dénombrement donnait-il naissance à des séries génératrices algébriques ? Comment expliquer les comportements observés, de nature différente selon les valeurs relatives de a et de b ? Enfin, était il possible de dériver des algorithmes de génération aléatoire linéaires pour ces objets, afin de permettre une évaluation de la sensibilité pour des séquences de grande taille ? C'est à ces aspects que nous nous sommes intéressés avec M. Bousquet-Mélou, donnant naissance aux résultats résumés dans le prochain chapitre.

11. PROPRIÉTÉS ÉNUMÉRATIVES ET COMPLEXITÉ DU REJET

Cette section décrit quelques observations, conjectures et résultats concernant l'énumération des chemins culminants. Soit $c_n^{a,b}$ le nombre de mots de taille n dans $\mathscr{C}^{a,b}$. Intuitivement, on va distinguer trois cas selon la *dérive* de la marche, définie comme la différence $a-b$. En effet, une marche aléatoire non contrainte de taille n ayant une dérive positive est réputée finir à une hauteur $\mathcal{O}(n)$ en moyenne et est intuitivement assez probablement positive et culminante. A l'opposé, une marche de dérive négative possède une très faible probabilité de rester positive. Nous avons examiné le cas particulier de la marche de dérive nulle, et obtenu des résultats énumératifs exacts dans la section 11.3.

Dans la suite du document, nous seront amenés à considérer le surensemble $\mathscr{M}^{a,b}$ des marches positives partant de $(0,0)$ et composées de pas $\{(1,a),(1,-b)\}$. L'énumération, à la fois exacte et asymptotique, de ces marches appelées *méandres discrets* dans la littérature probabiliste, et préfixes de mots de Dyck généralisés en combinatoire [54], a été complètement résolue dans [10]. Dans la suite du document, on notera $m_n^{a,b}$ le nombre de marches de taille n dans $\mathscr{M}^{a,b}$.

11.1 Marches de dérive positive ($a>b$) : Une conjecture

Quand la dérive est positive, on sait que la proportion des marches constituées de pas $+a$, $-b$ et restant positive est non-nulle. Plus précisément, quand $n\to\infty$,

$$m_n^{a,b}\sim\kappa_{a,b}.2^n$$

avec $\kappa_{a,b}$ une constante (explicite) positive. Nous pensons que la contrainte de culminance jouera un rôle *filtrant* similaire, et formulons la conjecture suivante

Conjecture 1 :
Quand $a>b$, il existe une constante $\kappa'_{a,b}$ telle que le nombre de chemins culminants de taille n est tel que

$$c_n^{a,b}\sim\kappa'_{a,b}.2^n.$$

Comme nous savons déjà que les méandres sont un surensemble des chemins culminants, et qu'un chemin culminant composé de pas $1,-1$ reste culminant si l'on remplace ses pas par $a,-b$, avec $a>b$, nous obtenons les bornes suivantes :

$$c_n^{1,1}\sim\frac{2^n}{4n}\leq c_n^{a,b}\leq m_n^{a,b}\sim\kappa_{a,b}2^n \tag{11.1}$$

L'asymptotique de $c_n^{1,1}$ repose sur le résultat de la section 11.3.

11.2 Marches de dérive négative ($a < b$) : Déficit exponentiel

Quand la dérive est négative, on sait que les marches positives sont exponentiellement rares parmi les marches aléatoires. En effet, Flajolet et al [10] montrent qu'il existe des constantes $\kappa_{a,b} > 0$ et $\alpha_{a,b} \in]1,2[$ telles que

$$m_n^{a,b} \sim \kappa_{a,b} \frac{\alpha_{a,b}^n}{n^{3/2}}.$$

, où $m_n^{a,b}$ est le nombre de marches positives sur des pas $\{(1,a),(1,-b)\}$, $b > a$. Plus précisément

$$\alpha_{a,b} = \frac{a+b}{\sqrt[a+b]{a^a b^b}} = \frac{1+q}{\sqrt[1+q]{q^q}} \equiv \alpha(q),$$

où $q = a/b < 1$. En dérivant $\alpha(q)$ selon q, on obtient

$$\alpha'(q) = -\frac{\ln(q)}{(1+q)\sqrt[1+q]{q^q}} > 0,$$

d'où l'on déduit que la constante de croissance exponentielle de l'asymptotique (Ou *ordre asymptotique*) de $m_n^{a,b}$ est une fonction croissante sur $q \in [0,1]$. De là, on déduit la borne supérieure suivante sur le nombre de chemins culminants :

$$c_n^{a,b} < m_n^{a,b} \sim \kappa_{a,b} \frac{\alpha_{a,b}^n}{n^{3/2}}, \quad \text{avec } \alpha_{a,b} \in]1,2[. \tag{11.2}$$

On en déduit qu'il n'existe qu'une proportion exponentiellement faible de marches de taille n et composée de pas $+a,-b$ qui soient culminants. Plus précisément, on présentera en section 11.7 la construction d'un langage $\mathscr{E}^{a,b}$, surensemble des chemins culminants, et qui nous permet de dériver la borne améliorée qui suit :

$$c_n^{a,b} \leq |\mathscr{E}_n^{1,1}| \in \mathscr{O}\left(\frac{\alpha_{a,b}^n}{n^3}\right). \tag{11.3}$$

Nous pensons poursuivre l'étude de ce cas, afin entre autre de déterminer si la constante de croissance des chemins culminants reste $\alpha_{a,b}$. On pourra pour cela utiliser le fait que la hauteur finale d'un méandre de longueur n suit une loi limite discrete quand $n \to \infty$.

11.3 Marches de dérive nulle ($a = b = 1$) : Résultats exacts

Pour la simplicité des notations, nous appellerons c_n le nombre de chemins culminants de taille n, quand $a = b = 1$. Soit $C(t) = \sum_{n>0} c_n t^n$ la série génératrice de dénombrement associée. Si un chemin culminant est de taille au moins égale à 2, alors nous pouvons supprimer le premier et le dernier pas, obtenant ainsi un chemin *faiblement culminant*, c'est à dire un chemin non-strictement positif dont le dernier pas atteint une hauteur supérieure ou égale à toutes celles précédemment atteintes. Soit $W(t)$ la série génératrice de dénombrement des chemins faiblement culminants. La construction de ces chemins implique que $C(t) = 1 + t + t^2 W(t)$.

Proposition 3 :
Soit $k \geq 0$. La série génératrice de dénombrement $W_k(t)$ des chemins faiblement culminants composés de pas $+1, -1$ est donnée par

$$W_k(t) = \frac{t^k}{F_{k+1}} = \frac{U^k(1-U^4)}{1-U^{2k+4}}, \tag{11.4}$$

où

$$U \equiv U(t) = \frac{1-\sqrt{1-4t^2}}{2t}$$

et F_k est le k-ième polynôme de Fibonacci, tel que $F_0 = F_1 = 1$ et $F_k = F_{k-1} - t^2 F_{k-2}$ pour $k \geq 2$. En conséquence, la série génératrice de dénombrement des chemins culminants en pas $+1, -1$ est

$$C(t) = \frac{1-U^2}{1+U^2} \sum_{k\geq 1} \frac{U^k}{1-U^{2k}}.$$

Quand $n \to \infty$, le nombre de chemins culminants en pas $+1, -1$ de taille n est équivalent asymptotiquement à $2^n/(4n)$. □

PREUVE : Commençons par l'énumération des chemins faiblement culminants de hauteur k. Tout d'abord, on dérive l'équivalence entre les expressions pour W_k du fait que

$$U = t(1+U^2) \quad \text{and} \quad F_k = \frac{1-U^{2k+2}}{(1-U^2)(1+U^2)^k}.$$

On peut prouver une telle expression pour F_k par récurrence sur k.

Ensuite, la façon la plus élégante de prouver l'équation (11.4) est probablement d'utiliser la correspondance établie par Viennot entre les marches dans un graphe et les empilements de cycles [99]. Résumons ici cette application. Cette correspondance se transpose en particulier en une bijection entre les chemins faiblement culminants de taille n et hauteur k, et les *empilements* de $(n-k)/2$ *dimers*, pris sur le segment $[0,k]$. Le *lemme d'inversion* (qu'on trouvera aussi dans [99]) nous dit que la série génératrice des empilements de dimers [99] sur ce segment est l'inverse de la série génératrice des *empilements triviaux* sur $[0,k]$, dont on peut facilement montrer qu'il s'avèrent être F_{k+1}. Le résultat s'ensuit.

Nous décrivons avec davantage de détails une approche alternative qui a le mérite d'être généralisée (jusqu'à un certain point) aux marches utilisant des pas $+a, -b$. Cette approche est basée sur une construction *pas à pas* des marches, et requiert de tenir compte d'un paramètre additionnel, la hauteur finale de la marche. Plus précisément, soit $G(t,u) \equiv G(u)$ la série génératrice des marches commençant en $(0,0)$ et restant confinées dans la bande comprise entre les hauteurs 0 et k, comptées sur leur longueur (variable t) et leur hauteur finale (variable u). Remarquons que $G(t,u)$ est une série formelle en t dont les coefficients sont polynômiaux en u. On peut aussi écrire $G(u) = \sum_{h=0}^{k} u^h G_h$, où G_h est la série génératrice de dénombrement des marches confinées finissant à hauteur k Alors G_k compte précisément les chemins faiblement culminants de taille k. De plus, une construction pas à pas de ces chemins donne

$$G(u) = 1 + t\left(u + \frac{1}{u}\right) G(u) - \frac{t}{u} G_0 - t u^{k+1} G_k,$$

c'est à dire,

$$\Big(u - t(1+u^2)\Big)G(u) = u - tG_0 - tu^{k+2}G_k. \tag{11.5}$$

On résout cette équation en utilisant la *méthode du noyau* [19; 9]. Le *noyau* de cette équation, c'est à dire le polynôme $K(t,u) = u - t(1+u^2)$, a un degré 2 en u. Ses deux racines sont

$$U_{0,1} = \frac{1 \mp \sqrt{1-4t^2}}{2t}.$$

Si l'on remplace u par U_0, puis par U_1, dans (11.5), on obtient alors deux équations linéaires reliant G_0 à G_k. En le résolvant puis en utilisant la relation $U_0U_1 = 1$, on trouve l'expression annoncée pour $G_k = W_k$ en fonction de $U_0 = U$. L'expression de $C(t)$ provient alors du fait que $C(t) = 1 + t + t^2W(t)$.

Résumons alors les étapes-clé de la dérivation du comportement asymptotique des c_n. On part de l'expression de $C(t)$, puis on utilise l'analyse de singularité de [40]. Étudions tout d'abord la partie *paire* de $C(t)$:

$$C_+(t) = \frac{1-U^2}{1+U^2}\sum_{k\geq 1}\frac{U^{2k}}{1-U^{4k}}.$$

Soit $Z \equiv Z(x)$ tel que $U(t)^2 = Z(t^2)$. C'est à dire,

$$Z \equiv Z(x) = \frac{1-2x-\sqrt{1-4x}}{2x}.$$

Alors $C_+(t) = D(t^2)$ pour

$$D(x) = \frac{1-Z}{1+Z}\sum_{k\geq 1}\frac{Z^k}{1-Z^{2k}},$$

Nous nous sommes donc ramené à l'étude du comportement asymptotique des coefficients de $D(x)$. Les séries $Z \equiv Z(x)$ ont pour rayon de convergence $1/4$. Elles sont analytiques en $\mathbb{C} \setminus [1/4, +\infty)$, avec exactement une singularité en $x = 1/4$. Son module maximum dans le disque $|x| \leq 1/4$ est atteint pour $x = 1/4$, et est égal à 1. Enfin, quand $x \to 1/4$,

$$Z(x) = 1 - 2\sqrt{1-4x} + O(1-4x).$$

En particulier, $|\arg(1-Z(x))| < \pi/4 + o(1)$. De plus, il existe un *camembert* de la forme

$$\Delta = \{x \neq 1/4 : |4x| < 1+\eta \text{ et } |\arg(1-4x)| > \phi\}$$

avec $\eta > 0$ et $0 < \phi < \pi/2$ où Z a un module strictement inférieur à 1.

Les séries

$$S(z) = \frac{1-z}{1+z}\sum_{k\geq 1}\frac{z^k}{1-z^{2k}}$$

ont rayon de convergence 1, and satisfont $S(Z(x)) = D(x)$. Cela implique que $D(x)$ est analytique dans le domaine Δ, et que nous pouvons envisager une *composition de singularités*, quand x est proche de $1/4$, et $Z(x)$ proche de 1.

Quand $z \to 1$ de telle façon que $|\arg(1-z)| < \alpha < \pi/2$ pour un certain α, alors

$$\sum_{k\geq 1} \frac{z^k}{1-z^{2k}} \sim \frac{1}{2(1-z)} \log \frac{1}{1-z}, \quad \text{tel que} \quad S(z) \sim \frac{1}{4} \log \frac{1}{1-z}.$$

Ceci peut être obtenu par une transformée de Mellin.

Par une composition des comportements en leur singularité de $Z(x)$ et $S(z)$, nous obtenons que, quand $x \to 1$ dans le domaine Δ,

$$D(x) \sim \frac{1}{8} \log \frac{1}{1-4x},$$

d'où nous concluons que le coefficient de x^n dans $D(x)$ est équivalent à $4^n/(8n)$. En transposant ce résultat sur les séries $C_+(t)$, on obtient que le nombre de chemins culminants de taille paire $N = 2n$ est équivalent à $2^N/(4N)$. L'étude de la partie impaire $C_-(t)$ est similaire. ∎

Remarque : La construction *pas à pas* des chemins dans une bande de hauteur k, combinée avec la méthode du noyau, nous permet de calculer la série génératrice des chemins (faiblement) culminants pour toute valeur (donnée) de a et b. Cette série génératrice peut être exprimée comme une simple fonction rationnelle en les racines U_i d'un polynôme calculable. Cette fonction a donc pu être établie explicitement pour des valeurs $(1,-2)$ et $(2,-1)$ de a et b. Il reste à dériver de ces résultats le comportement asymptotique du nombre de chemins culminants généraux.

11.4 Efficacité de la méthode par rejet

On s'intéresse ici à une autre famille d'algorithmes de génération aléatoire, employant une technique dite *de rejet*, principe décrit en section 3.3.2 ou en [31]. Le principe extrêmement simple d'un algorithme *par rejet* pour les des mots de $\mathcal{W}$ consiste à engendrer des mots uniformément dans un surensemble $\mathcal{V} \supset \mathcal{W}$ jusqu'à ce qu'un objet de $\mathcal{W}$ soit obtenu. La complexité d'une telle technique est alors $\zeta(n)\mathrm{w}_n/\mathrm{v}_n$, où $\zeta(n)$ est le coût de génération d'un mot de taille n dans $\mathcal{V}$, et w_n et v_n sont les nombres de mots de taille n dans $\mathcal{W}$ et $\mathcal{V}$.

La complexité de cette méthode repose alors sur celle de la génération dans $\mathcal{V}$. On peut cependant, en relâchant des contraintes, obtenir des algorithmes efficaces. Par exemple, on remarquera que le langage $\{m, \overline{m}\}^n$ admet un générateur efficace en un tirage de *n pile ou face*, et que ce langage est un surensemble des chemins positifs (eux mêmes surensembles des chemins culminants).

L'intérêt de cette méthode est principalement l'absence de surcoût en mémoire, ainsi que l'absence d'étape de précalcul. Cependant, la génération de mots de $\mathcal{V}$ est d'une certaine façon cachée dans la description ci-dessus, et est susceptible d'impliquer un algorithme consommateur en temps. De plus, le test d'appartenance des objets engendrés à $\mathcal{W}$ pourrait, en tout généralité, provoquer un surcoût à prendre en compte dans la complexité générale des algorithmes présentés ci-dessous. Il n'en est rien ici, car la positivité et la culminance d'une marche peuvent être vérifiées en une unique passe sur le codage de celle-ci.

Nous examinons donc dans la suite plusieurs candidats pour $\mathcal{V}$.

11.5 Rejet à partir des marches non-contraintes

$$\mathcal{V} := \{m, \overline{m}\}^*$$

Ce choix peut sembler extrêmement naïf, mais s'avère raisonnable quand $a \geq b$ (dérive positive). Les mots de taille n dans $\mathcal{V}$ peuvent être engendrés en tirant à *pile ou face n* fois, ce qui implique $\zeta(n) = n$ et $v_n = 2^n$, dans les notations ci-dessus. Le coût moyen de l'algorithme de rejet est donc $n2^n/c_n^{a,b}$, qu'on peut estimer grâce aux résultats et conjectures de la section 11.

Quand $a < b$, alors l'équation (11.2) révèle une complexité exponentielle de constante de croissance $2/\alpha_{a,b}$.

Quand $a = b$, la proposition 3 implique une complexité en $\mathcal{O}(n^2)$.

Enfin, quand $a > b$, la conjecture 1 suggère que cet algorithme est linéaire, et sûrement bornée par une complexité quadratique.

11.6 Rejet à partir des méandres

$$\mathcal{V} := \mathcal{M}^{a,b}$$

Les méandres $\mathcal{M}^{(a,b)}$, définis en section 11, peuvent être engendrés des deux façons suivantes : Une décomposition *dernier passage* prudemment adaptée de [35] permet de dériver une approche récursive en $\mathcal{O}(n\log(n))$ après un précalcul en $\mathcal{O}(n^2)$, qu'on utilisera quand $a < b$; Une technique de rejet anticipé [11] nous donne une complexité en $\mathcal{O}(n)$ quand $a > b$.

Quand $a < b$, le nombre v_n de méandres de taille n se comporte comme $\alpha_{a,b}^n n^{-3/2}$, à une constante multiplicative près. Si $c_n^{a,b}$ se comporte comme $\alpha_{a,b}^n n^{-3-\gamma}$ pour $\gamma \geq 0$ (voir l'équation (11.2)), la complexité sera en $\mathcal{O}(n^{\gamma+5/2}\log n)$, après un précalcul en $\mathcal{O}(n^2)$ (sans parler de la construction de la grammaire reconnaissant les méandres). Si la constante de croissance de $c_n^{a,b}$ est inférieure à $\alpha_{a,b}$, alors la complexité sera exponentielle.

Si $a = b$, alors v_n se comporte comme $2^n/\sqrt{n}$, alors que $c_n \sim 2^n/n$, d'où une complexité en $\mathcal{O}(n^{3/2})$ (A une constante multiplicative près).

11.7 Utilisation du langage des images-miroir

Proposition 4 :
Soit $\overline{\omega}$ l'image miroir d'un mot ω. Alors

$$\omega \in \mathscr{C}^{a,b} \Leftrightarrow \overline{\omega} \in \mathscr{C}^{a,b}.$$

La preuve de cette propriété repose sur le fait que les conditions de positivité et de culminance jouent des rôles symétriques. La transformation correspondant graphiquement à une symétrie centrale sur la marche, ces rôles sont alors échangés, et la marche obtenue est bien un chemin culminant.

Un corollaire de cette propriété est qu'en moyenne, la marche atteint la moitié de sa hauteur finale en son milieu. De plus, on peut sentir intuitivement que la condition de positivité sera surtout discriminante au début de la marche, et la condition de culminance principalement à la fin de celle-ci. La proposition 4 nous permettant désormais d'engendrer efficacement des chemins culminants non-nécessairement positifs (image miroir d'un méandre), ce qui nous permet d'envisager un nouveau surensemble dans lequel pratiquer la génération par rejet : Le langage $\mathscr{E}^{a,b}$ des mots positifs dans leur première moitié, et culminants dans leur deuxième. Plus précisément

$$\mathscr{E}^{a,b} := \bigcup_{n\geq 0} \mathscr{M}^{a,b}_{\lceil \frac{n}{2} \rceil} \overline{\mathscr{M}^{a,b}_{\lfloor \frac{n}{2} \rfloor}},$$

où $\mathscr{M}^{a,b}$ est le langage des méandres en pas $+a, -b$, et $\overline{\mathscr{M}^{a,b}}$ est le langage des images-miroirs des mots de $\mathscr{M}^{a,b}$. Il est alors évident que $\mathscr{C}^{a,b} \subset \mathscr{E}^{a,b}$. Étudions maintenant l'efficacité de la méthode de rejet basée sur ce langage.

Quand $a = b$,

$$|\mathscr{E}^{1,1}_n| = |\mathscr{M}^{1,1}_{\lceil \frac{n}{2} \rceil}||\overline{\mathscr{M}^{1,1}_{\lfloor \frac{n}{2} \rfloor}}| \sim |\mathscr{M}^{1,1}_{\frac{n}{2}}|^2 \sim \kappa^2_{1,1} \left(\frac{2^{n/2}}{\sqrt{\frac{n}{2}}} \right)^2 \sim 2\kappa^2_{1,1} \frac{2^n}{n}.$$

Comme on sait générer les méandres en temps $\mathscr{O}(n)$, et que $c^{1,1}_n \sim 2^n/n$ (à une constante multiplicative près), alors on en déduit que la complexité de la méthode de rejet basée sur $\mathscr{E}^{1,1}$ est *linéaire*.

Enfin, pour $a < b$, on a $|\mathscr{E}^{a,b}_n| \sim \alpha^n_{a,b}/n^3$, car $m^{a,b}_n \sim \alpha^n_{a,b}/n^{3/2}$, et le gain obtenu par rapport au tirage *pile ou face* est donc de l'ordre de $\mathscr{O}(n^{3/2})$.

11.8 Algorithmes

Dans le **cas d'une dérive négative** ($a < b$), aucune optimisation de l'approche récursive ne résulte de l'étude des propriétés énumératives des chemins culminants.

Dans le **cas d'une dérive nulle** ($a = b$), un algorithme en temps linéaire sans précalcul est le suivant, pour des séquences de tailles attendues n :

1) Engendrer aléatoirement uniformément des chemins positifs (préfixes de Dyck) α et β de tailles respectives $\lceil \frac{n}{2} \rceil$ et $\lfloor \frac{n}{2} \rfloor$.

2) Affecter $\omega := \alpha\overline{\beta}$, où $\overline{\beta}$ est l'image miroir de β.

3) Si $\omega \in \mathscr{C}^{1,1}$, alors retourner ω.
Sinon retourner à l'étape 1).

La complexité linéaire proclamée repose sur la linéarité de chacune des trois étapes ci-dessus, ainsi que sur le faible nombre, constant en moyenne, de rejets 3)→1). En effet, l'étape 1) peut être réalisée en temps linéaire, par une technique de rejet anticipé par exemple [11]. L'étape 2) revient à une copie de chaîne de caractère, et l'étape 3) peut être réalisée en une unique passe, en lisant la séquence candidate de gauche à droite, vérifiant à chaque nouveau pas la positivité du préfixe et gardant en mémoire l'altitude maximale atteinte en vue d'une comparaison à l'altitude finale. Le nombre de rejets 3)→1) est quant à lui constant en moyenne grace au rapport constant des cardinaux des ensembles $\mathscr{E}^{1,1}$ et $\mathscr{C}^{1,1}$. La complexité de cette génération est donc bien linéaire.

Dans le **cas d'une dérive positive** ($a > b$), un algorithme en temps linéaire sans précalcul est encore plus simple, en tenant pour acquise la conjecture 1 :

1) Engendrer uniformément $\omega \in \{m, \overline{m}\}^n$.

2) Si $\omega \in \mathscr{C}^{a,b}$, alors retourner ω.
Sinon retourner à l'étape 1).

Comme explicité ci-dessus, la génération de l'étape 1) est linéaire ($\Leftrightarrow n$ bits aléatoires) le test d'appartenance de ω à $\mathscr{C}^{a,b}$ est linéaire, et le nombre de rejets 2)→1) constant en moyenne, par un argument de cardinalité. La complexité de cet algorithme est donc elle aussi linéaire.

11.9 Conclusions et perspectives

Nous avons étudié les chemins culminants, à la fois des points de vue langages formels, génération aléatoire et combinatoire. En utilisant une application de la méthode récursive, nous avons obtenu un algorithme de génération aléatoire linéaire après un précalcul en $\mathscr{O}(n^3)$ opérations arithmétiques et espace. Une étude des propriétés énumératives de ces chemins a permis en outre d'évaluer l'efficacité de la génération par rejet, pratiquée à partir de trois classes de langages différents, dérivant les complexités de la table 11.1. On notera que les complexités marquées d'une étoile * dénotent un nombre d'opérations arithmétiques, potentiellement inférieur au temps réel d'exécution de l'algorithme d'un facteur n, en raison de la taille, exponentielle sur n, des nombres manipulés (voir section 3.3.1). Dans le cas $a > b$, la complexité linéaire de l'approche par rejet repose sur une conjecture formulée sur le nombre de chemins culminants, que nous prévoyons de prouver dans un futur proche.

Des extensions futures des méthodes présentées ici pourraient prendre en charge différent types de pas montants $(+1,+m_1),\ldots,(+1,+m_u)$ et descendants $(+1,+\overline{m}_1),\ldots,(+1,+\overline{m}_d)$, correspondant à des bonus/pénalités associés à des bases traditionnellement conservées, capturant ainsi intégralement le système de score de l'algorithme FLASH. Il semblerait alors que le pendant de la dérive soit la différence $\Sigma_{i=1}^{u} m_i - \Sigma_{i=1}^{u} \overline{m}_i$, et qu'on retrouve les comportements étudiés ici.

	Méthode	Essais	Précalcul	Coût/Séquence
$\mathscr{C}^{a,b\Rightarrow k}$	Récursive (version rationnelle)	1	$\mathcal{O}(nk)^*$	$\mathcal{O}(n)$
$a<b$	Récursive [53]	1	$\mathcal{O}(n^3)^*$	$\mathcal{O}(n\log(n))$
$a=b$	Rejet (Images mirroir)	$\mathcal{O}(1)$	$\varnothing$	$\mathcal{O}(n)$
$a>b$	Rejet	$\underset{\text{conj}}{\sim}\ \mathcal{O}(1)$ $\underset{\text{déf.}}{\leq}\ \mathcal{O}(n)$	$\varnothing$	$\underset{\text{conj.}}{\sim}\ \mathcal{O}(n)$ $\underset{\text{def.}}{\leq}\ \mathcal{O}(n^2)$

Tab. 11.1: Domaines de validité des différentes approches

De plus, il peut être intéressant, afin de capturer d'évaluer la sensibilité de façon encore plus réaliste, de tenir compte d'un modèle probabiliste additionnel, comme l'ont fait Keich et al dans [52] pour un modèle de Bernoulli. Ici, nous pourrions facilement injecter un modèle de Markov dans l'approche récursive, en comptant les sommes des probabilités des séquences ayant un certain préfixe plutôt que le nombre de ces séquences. Le surcoût engendré est alors en α^k, où k est l'*ordre* du modèles. Il est aussi possible d'utiliser les probabilités de transition du modèle au cours de la génération dans l'approche par rejet. Cependant, une telle transposition mérite une étude minutieuse, dans la mesure où :

- **Correction :** L'approche par *images-miroir* dans le cas $a=b$ ne doit pas faire l'objet d'une adaptation irraisonnée, dans la mesure où il est hautement improbable (de l'ordre de $\mathcal{O}(1/\alpha^{2k})$) que les deux demi-séquences finissent dans des états *symétriques*. Cette condition est pourtant nécessaire au respect de la distribution markovienne. Cependant, un rejet à partir des méandres est susceptible de résoudre ce problème, au prix d'une complexité $\mathcal{O}(n\sqrt{n})$.

- **Efficacité :** Au cours de la génération, la prise en compte de probabilités markoviennes a clairement une influence sur la dérive *réelle* de la marche. Comme nous l'avons déjà remarqué, le nombre de méandres pourrait alors être exponentiellement plus faible que 2^n, résultant en une complexité exponentielle pour le rejet, et ce même dans le cas $a>b$. Peut être devrait on alors retrouver nos différents régimes en s'intéressant à des produits pondérés par les probabilités d'émission au moment d'évaluer la dérive, comme le fait Louchard dans [57] ? Par exemple, dans un modèle de Bernoulli attribuant p_m à une conservation m, et $p_{\overline{m}}=1-p_m$ à une altération $\overline{m}$, la dérive considérée pourrait être la différence $ap_m-bp_{\overline{m}}$.

Enfin, on envisage d'étendre les résultats énumératifs exacts aux cas $(+1,-2)$ et $(+2,-1)$. Une telle extension nous fournirait en effet une intuition sur la valeur réelle de la constante de croissance dans le cas $a<b$, et corroborerait notre conjecture pour le cas $a>b$.

12. CONCLUSION ET PERSPECTIVES

12.1 Conclusion

Au cours de cette thèse, on s'est intéressé à la génération aléatoire de séquences structurées. De tels modèles de séquences ont justifié l'étude du formalisme des grammaires pondérées, ainsi que son ajout au sein du logiciel `GenRGenS`. La dérivation automatique de pondérations permettant l'*imitation* de séquences réelles a été étudiée, à la fois d'un point de vue analytique et à travers l'implémentation d'un logiciel basé sur une approche d'*optimisation*. Ce logiciel a alors été appliqué à la génération aléatoire de structures secondaires d'ARN, des objets biologiques particulièrement bien modélisés par des grammaires pondérées. Un algorithme quadratique de planarisation de la structure d'ARN a alors été proposé, afin de tirer un parti optimal des récentes avancées dans l'inférence des liaisons d'un ARN dont on connaît une conformation. Enfin, dans des travaux conjoints avec G. Kucherov et L. Noé, puis M. Bousquet-Mélou, nous avons introduit et étudié les chemins culminants, une variété de marche aléatoire qui modélise le cheminement du score dans un algorithme heuristique de recherche de similarité. L'étude poussée de ces chemins a permis la découverte d'algorithmes de génération aléatoire efficaces, basés sur des décompositions qu'on a pu adapter pour évaluer la sensibilité des algorithmes heuristiques de recherche de similarités. Les propriétés combinatoires de ces chemins se sont avérées assez surprenantes, faisant apparaître des séries génératrices transcendantes et des langages contextuels. Enfin, les nombreuses facettes de l'approche *récursive* pour la génération aléatoire nous ont laissé entrevoir la possibilité d'étendre ce formalisme. On a donc proposé une approche récursive *généralisée*, qui a comme mérite d'autoriser des transformations morphiques ainsi que très peu de restrictions dans la récurrence, ce qui permet des incursions dans l'univers contextuel.

12.2 Perspectives : `GenRGenS`

`GenRGenS`devrait, au cours des suites données à cette thèse, faire l'objet d'un travail organisé autour de trois axes :

1. *Réflexion autour de la notion de modèle de séquence* : Les modèles de séquences sont omniprésents au sein des méthodes bioinformatiques. Cependant, ceux-ci sont souvent un peu trop simples, et peinent à capturer l'impact de phénomènes connus et admis. Idéalement, il faudrait être capable de modéliser tous les phénomènes déjà avérés, dans nos séquences d'intérêts, avant d'utiliser des séquences aléatoires pour valider la significativité d'un nouveau phénomène. Mais, alors que les connaissances s'accumulent sur certaines portions du génomes, est-il possible d'adopter une approche *séquentielle* pour la création du modèle ? Un nouveau modèle pourrait alors être construit à partir du précédent et du phénomène nouvellement compris. C'est dans cette optique que les modèles *hiérarchiques* ont été ajoutés à `GenRGenS`. Cependant, de tels modèles ne risquent-ils alors pas d'interagir pour donner naissance à des phénomènes *émergeants* ? Existe-t'il des classes de modèles *orthogonaux*, c'est à dire tels que les propriétés du modèle *produit* soit exactement la sommes des propriétés des modèles ? De telles investigations sont nécessaires pour permettre une construction itérative des modèles de séquences structurées.

2. Amélioration des algorithmes et modèles : La présente version de `GenRGenS`ne tient pas encore compte de toutes les améliorations susceptibles d'être apportées à la génération de mots d'un langage non-contextuel. De plus, son codage en Java fait perdre des constantes non-négligeables à celui qui souhaite engendrer des séquences de grande taille. Il est donc dans les perspectives de mes travaux de continuer le portage de `GenRGenS`dans un langage compilé, comme le C++. Enfin, la construction proposée pour le produit de modèles Grammaire×Markov est loin d'être optimale, on pourrait donc implémenter soigneusement un tel algorithme.

3. Intégration de nouveaux modèles : `GenRGenS`a pour vocation d'intégrer le plus grand nombre de familles de générateurs possible. On s'intéressera tout d'abord aux grammaires stochastiques. Prises comme un mécanisme de génération aléatoire, pendants algébriques des HMM pour les séquences, celles-ci ne se révèlent guère intéressantes (Pas de garantie de convergence). Cependant, on a vu que la génération de Boltzmann utilise une structure algorithmique proche des grammaires stochastiques, consacrant toutes ses probabilités de réécritures à l'obtention d'une complexité linéaire en taille approchée. De plus, F. Weinberg adapte en [102] la stratégie récursive aux grammaires stochastiques, un précalcul permettant la génération d'une séquence de taille souhaitée avec une probabilité égale à celle du modèle initial, renormalisée. Peut être ce formalisme pourrait-il être intégré à GenRGenS dans un proche futur.

   Enfin, on s'intéressera aux modèles de *shuffling*, ou mélange, qui permettent de respecter les nombres de motifs d'une taille donnée de la séquence originale dans la séquence engendrée. Ces modèles ont fait l'objet d'extensions imposant des contraintes supplémentaires, comme l'interdiction de certains motifs [80], et sont alors particulièrement adaptés à la modélisation de l'ADN. On prévoit l'intégration du logiciel SMACK, dû à R. Rivière et implémentant ces principes, dans GenRGenS.

12.3 Perspectives : ARN et structures

12.3.1 ARN réalistes

Tout d'abord, on tentera d'étendre les calculs de pondérations à des valeurs pour les paramètres calculées dans toutes les familles d'ARN. Une fois ces poids obtenus, on peut s'en servir pour classifier les ARN. En particulier, on voudrait comparer les valeurs pour les paramètres structuraux dans les bases de données de Gutell [22] et de la NDB [15] (Méthodologie : RNAView+Planarisation).

De plus, il est presque sûr que les résultats analytiques peuvent être étendus au cas où l'on souhaite uniquement contrôler la taille moyenne des hélices. Dans des cas plus généraux, une résolution analytique restera probablement problématique, c'est pourquoi il est important d'étudier les *paysages* dessinés par la fonction objectif. En particulier, existe-t-il un unique minimum local (vallée) ? Comment expliquer les différentes vitesses de convergence constatées ? Existe-t-il des *transitions de phase* dans ce problème ?

12.3.2 Planarisation

L'algorithme ayant fait l'objet d'une implémentation, il est donc possible d'analyser massivement les bases de données structurelles, pour évaluer la proportion de base dans des pseudo-noeuds. Une réflexion sur ce qu'est un pseudo-noeud, c'est à dire incidemment sur la définition de la structure secondaire, devra alors être menée, de pair avec des biochimistes, biologistes et algorithmiciens.

De plus, on voudrait ajouter le modèle d'énergie libre de Turner aux critères à maximiser dans le cadre de la planarisation, afin par exemple d'approximer la proportion d'énergie maximale contenue dans une structure, une fois les pseudo-noeuds extraits.

12.4 Perspectives : Chemins culminants

Les chemins culminants ont fait l'objet de recherches assez poussées maintenant. Il nous reste cependant encore quelques conjectures à prouver, et des constantes à évaluer. On pourrait aussi s'intéresser à l'impact d'un modèle de Markov sur les constructions pas à pas utilisées pour les algorithmes de rejet. Se pourrait il que l'expression de la dérive fasse appel à la probabilité stationnaire de la chaîne de Markov ? De plus, cette probabilité stationnaire sera-t-elle influencée par les conditions de positivité et de culminance ? Existe-t-elle même encore ?

De même pour l'adjonction de différents types de pas négatifs correspondant aux différents types d'altérations. Qu'en serait-il alors de l'impact sur la dérive, ou ce qui en tiendrait lieu dans cette nouvelle marche ?

Enfin, la généralisation de l'approche récursive pourrait bénéficier d'au moins une amélioration *générique*, et dont pourraient bénéficier toutes les décomposition dans lesquelles le degré d'un noeud n'est pas borné. En effet, on peut distinguer la contribution de la taille à l'espace d'état, ce paramètre existant toujours dans nos objets d'intérêt. On peut alors s'intéresser aux noeuds conjonctifs et implémenter la stratégie *boustrophédon* pour gagner un facteur $\mathcal{O}(\frac{n}{\log(n)})$.

BIBLIOGRAPHIE

[1] S. E. Hambrusch A. Apostolico, M. J. Atallah. New clique and independent set algorithms for circle graphs. *Discrete Applied Mathematics*, 36(1):1–24, 1992.

[2] D. Abergel. *Caractérisation bioinformatique des régions inter-ORF chez la levure*. PhD thesis, Université Paris Sud, Ecole doctorale Gènes, Génomes et Cellules, December 2004.

[3] J. Allali and M. F. Sagot. A multiple graph layers model with application to RNA secondary structures comparison. In *String Processing and Information Retrieval 2005*, volume 3772, pages 348–359, 2005.

[4] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

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

[6] M. D. Atkinson and J.-R. Sack. Generating binary trees at random. *Information Processing Letters*, 41(1):21–23, 1992.

[7] R. K. Azad and J. G. Lawrence. Use of artificial genomes in assessing methods for atypical gene detection. *PLoS Computational Biology*, 1(6):e56, November 2005.

[8] A. Bairoch. The PROSITE dictionary of sites and patterns in proteins, its current status. *Nucleic Acids Research*, 13(21):3097–3103, 1993.

[9] C. Banderier, M. Bousquet-Mélou, A. Denise, P. Flajolet, D. Gardy, and D. Gouyou-Beauchamps. Generating functions for generating trees. *Discrete Math.*, 246(1-3):29–55, 2002.

[10] C. Banderier and P. Flajolet. Basic analytic combinatorics of directed lattice paths. *Theor. Comput. Sci.*, 281(1-2):37–80, 2002.

[11] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of underdiagonal walks. In Pierre Leroux and Christophe Reutenauer, editors, *Proceedings of 4th Conference on Formal Power Series and Algebraic Combinatorics (FPSAC'92)*. Université du Québec à Montréal, 1992.

[12] F. Bassino and C. Nicaud. Enumeration and random generation of accessible automata. *Soumis...*, 2006. Preprint disponible à http://www-igm.univ-mlv.fr/ bassino/publications/tcs06.ps.

[13] M. Bekaert. *Étude du décalage de phase de lecture dans le génome de Saccharomyces cerevisiæ*. PhD thesis, Université Pierre et Marie Curie, Paris VI, November 2004.

[14] F. Vanden Berghen. *CONDOR: a constrained, non-linear,derivative-free parallel optimizer for continuous, high computing load, noisy objective functions*. PhD thesis, IRIDIA, Université Libre de Belgique, 2005.

[15] H. M. Berman, W. K. Olson, D. L. Beveridge, J. Westbrook, A. Gelbin, T. Demeny, S. H. Hsieh, A. R. Srinivasan, and B. Schneider. The nucleic acid database. a comprehensive relational database of three-dimensional structures of nucleic acids. *Biophysical Journal*, 63(3):751–759, 1992.

[16] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Researches*, 28(1):235–242, 2000.

[17] G. Blin, G. Fertin, R. Rizzi, and S. Vialette. What makes the arc-preserving subsequence problem hard ? In *Proceedings of the 1st International Workshop on Bioinformatics Research and Applications (IWBRA'05)*, volume 3515 of *Lecture Notes in Computer Science*, pages 860–868. Springer-Verlag, 2005.

[18] G. Blin, G. Fertin, and S. Vialette. New results for the 2-interval pattern problem. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM'04)*, volume 3109 of *Lecture Notes in Computer Science*, pages 311–322. Springer-Verlag, 2004.

[19] M. Bousquet-Mélou and M. Petkovšek. Linear recurrences with constant coefficients: the multivariate case. *Discrete Math.*, 225(1-3):51–75, 2000.

[20] S. Burkhardt and J. Kärkkäinen. Better filtering with gapped q-grams. *Fundamenta Informaticae.*, 56(1–2):51–70, 2003.

[21] A. Califano and I. Rigoutsos. Flash: A fast look-up algorithm for string homology. In *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, pages 56–64. AAAI Press, 1993.

[22] 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. Muller, 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. *BioMed Central Bioinformatics*, 3(2), 2002.

[23] S. Cao and S.J. Chen. Predicting rna pseudoknot folding thermodynamics. *Nucleic Acids Research*, 34(9):2634–2652, 2006.

[24] A. Condon, B. Davy, B. Rastegari, S. Zhao, and F. Tarrant. Classifying rna pseudoknotted structures. *Theoretical Computer Science*, 320(1):35–50, 2004.

[25] E. Coward. Shufflet: Shuffling sequences while conserving the k-let counts. *Bioinformatics*, 15(12):1058–1059, 1999.

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

[27] A. Denise. *Structures aléatoires, modèles et analyse des génomes*. PhD thesis, Laboratoire de Recherche en Informatique-, 2001. Mémoire d'habilitation à diriger des recherches.

[28] A. Denise, Y. Ponty, and M. Termier. Random generation of structured genomic sequences. In *Proceedings of RECOMB 03(Berlin)*, 2003. Poster.

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

[30] A. Denise and P. Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theor. Comput. Sci.*, 218(2):233–248, 1999.

[31] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[32] Y. Ding, C. Y. Chan, and C. E. Lawrence. Rna secondary structure prediction by centroids in a boltzmann weighted ensemble. *RNA*, 11:1157–1166, 2005.

[33] Y. Ding and E. Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Research*, 31(24):7280–7301, 2003.

[34] M. Drmota. Systems of functional equations. *Random Structures and Algorithms*, 10(1-2):103–124, 1997.

[35] P. Duchon. On the enumeration and generation of generalized Dyck words. *Discrete Maths*, 225(1-3):121–135, 2000.

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

[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] S. R. Eddy and R. Durbin. Rna sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994.

[39] P. Flajolet, E. Fusy, and C. Pivoteau. Boltzmann sampling of unlabelled structures. *Soumis...*, 2006. Preprint disponible à http://algo.inria.fr/flajolet/Publications/FlFuPi06.pdf.

[40] P. Flajolet and A. Odlyzko. Singularity analysis of generating functions. *SIAM J. Discrete Math.*, 3(2):216–240, 1990.

[41] P. Flajolet, P. Zimmermann, and B. Van Custem. A calculus for the random generation of combinatorial structures. Technical Report RR-1830, INRIA, 1993.

[42] P. Flajolet, P. Zimmermann, and B. Van Cutsem. Calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132:1–35, 1994. A preliminary version is available in INRIA Research Report RR-1830.

[43] E. Fusy. Quadratic exact-size and linear approximate-size generation of planar graphs. In *International Conference on Analysis of Algorithms*, volume AD, pages 125–138, 2005.

[44] D. Gautheret and A. Lambert. Direct rna motif definition and identification from multiple sequence alignments using secondary structure profiles. *Journal of Molecular Biology*, 313:1003–1011, 2001.

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

[46] C. Herrbach, A. Denise, S. Dulucq, and H. Touzet. Alignment of rna secondary structures using a full set of operations. Technical Report 1451, Université Paris-Sud 11, 2006.

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

[48] J. E. Hopcroft and J. D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley, 1969.

[49] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata and Language Theory*. Addison-Wesley, 1979.

[50] R. Hugues. *Prédiction de la localisation cellulaire des protéines à l'aide de leurs séquences biologiques*. PhD thesis, Université d'Evry Val d'Essonne, Mathématiques appliquées, December 2005.

[51] T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between rna structures. *Journal Computational Biology*, 9(2):371–388, 2002.

[52] U. Keich, M. Li, B. Ma, and J. Tromp. On spaced seeds for similarity search. *Discrete Applied Mathematics*, 138(3):253–263, 2004.

[53] G. Kucherov, L. Noe, and Y. Ponty. Estimating seed sensibility on homogenous alignments. In IEEE, editor, *Proceedings of Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'04)*, page 387, 2004.

[54] J. Labelle and Y.N. Yeh. Generalized Dyck paths. *Discrete Math.*, 82(1):1–6, 1990.

[55] S. Lemieux and F. Major. Rna canonical and non-canonical base pairing types: a recognition method and complete repertoire. *Nucleic Acids Research*, 30(19):4250–4263, 2002.

[56] N. Leontis and E. Westhof. Geometric nomenclature and classification of rna base pairs. *RNA*, 7:499–512, 2001.

[57] G. Louchard. Asymptotic properties of some underdiagonal walk generation algorithms. *Theoretical Computer Science*, 218:935–954, 1999.

[58] R. B. Lyngsø and C. N. S. Pedersen. Rna pseudoknot prediction in energy-based models. *Journal of Computational Biology*, 7(3-4):409–427, 2000.

[59] B. Ma, J. Tromp, and M. Li. Patternhunter: Faster and more sensitive homology search. *Bioinformatics*, 18(18):440–445, 2002.

[60] T.J. Macke, D.J. Ecker, R.R. Gutell, D. Gautheret, D.A. Case, and R. Sampath. Rnamotif, an rna secondary structure definition and search algorithm. *Nucleic Acids Research*, 29(22):4724–4735, 2001.

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

[62] D.H. Mathews and D.H. Turner. Prediction of rna secondary structure by free energy minimization. *Current Opinion in Structural Biology*, 16(3):270–278, 2006.

[63] V. L. Murthy and G.D. Rose. Rnabase: an annotated database of rna structures. *Nucleic Acids Research*, 31(1):502–504, 2003.

[64] M. Nebel. Combinatorial properties of rna secondary structures. *Journal of Computational Biology*, 3(9):541–574, 2003.

[65] M.E. Nebel. Identifying good predictions of rna secondary structure. In *Pacific Symposium on Biocomputing*, volume 9, pages 423–434, 2004.

[66] P. Nicodeme, B. Salvy, and P. Flajolet. Motif statistics. In *Proc. European Symposium on Algorithms-ESA'99, Prague*, pages 194–211, 1999.

[67] R. Nussinov and A.B. Jacobson. Fast algorithm for predicting the secondary structure of single-stranded rna. *Proc Natl Acad Sci U S A*, 77:6903–13, 1980.

[68] K. Panagiotou and A. Weißl. Properties and sampling of tree-like structures with blocks of higher connectivity. *Soumis...*, 2006. Preprint disponible à http://www.ti.ethz.ch/as/people/panagiotou/papers/oapts.ps.

[69] Laurent Tichit Pascal Ferraro and Serge Dulucq. Local similarity between trees. In *Acte de JOBIM'04*, 2004.

[70] Amy E. Pasquinelli, Brenda J. Reinhart, Frank Slack, Mark Q. Martindale, Mitzi I. Kuroda, Betsy Maller, David C. Hayward, Eldon E. Ball, Bernard Degnan, Peter Müller, Jürg Spring, Ashok Srinivasan, Mark Fishman, John Finnerty, Joseph Corbo, Michael Levine, Patrick Leahy, Eric Davidson, and Gary Ruvkun. Conservation of the sequence and temporal expression of let-7 heterochronic regulatory rna. *Nature*, 408:86–89, 2000.

[71] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 85, pages 2444–2448, 1988.

[72] Y. Ponty. GenRGenS : Generation of random genomic sequences. In *Actes des Journées Ouvertes Biologie Informatique et Mathématiques(JOBIM 02) à St Malo*, 2002.

[73] Y. Ponty. Etudes combinatoire et génération aléatoire des structures secondaires d'arn. Master's thesis, Université Paris Sud, 2003. Mémoire de DEA.

[74] Y. Ponty and A. Denise. Genrgens v2.0 user manual. Technical Report 1447, Rapport de recherche du Laboratoire de Recherche en Informatique (LRI), April 2006.

[75] Y. Ponty, M. Termier, and A. Denise. Generating random genomic sequences and structures with genrgens. In *Actes des Journées Ouvertes de Biologie, Informatique et Mathématiques - JOBIḾ06*, July 2006.

[76] Y. Ponty, M. Termier, and A. Denise. Genrgens: Software for generating random genomic sequences and structures. *Bioinformatics*, 22(12):1534–1535, 2006.

[77] J. Reeder and R. Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5:104, 2004.

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

[79] E. Rivas and S.R. Eddy. A dynamic programming algorithm for rna structure prediction including pseudoknots. *Journal of Molecular Biology*, 285:2053–2068, 1999.

[80] R. Rivière. *Algorithmes de graphes pour l'analyse des séquences et structures génomiques*. PhD thesis, Université Paris-Sud, Ecole doctorale d'informatique, September 2005.

[81] R. Rivière, D. Barth, J. Cohen, and A. Denise. Shuffling biological sequences with motifs constraints. In *Proc 1st Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets 2004)*. KCL publications, 2004.

[82] H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[83] B. Salvy and P. Zimmerman. Gfun: a maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Softwares*, 20(2):163–177, 1994.

[84] G. Schaeffer and D. Poulalhon. Counting, coding and sampling with words. In M. Lothaire, editor, *Applied Combinatorics on Words*, pages 443–478. Cambridge University Press, 2005.

[85] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. (German) [Fast multiplication of large numbers]. *Computing*, 7(3–4):281–292, 1971.

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

[87] Y.J. Sheng, Y.C. Mou, and H.K. Tsao. Conformational entropy of a pseudoknot polymer. *Journal of Chemical Physics*, 124(12):124904, 2006.

[88] H. Shi and P.B. Moore. The crystal structure of yeast phenylalanine trna at 1.93 a resolution: a classic structure revisited. *RNA*, 6(8):1091–1105, 2000.

[89] C. Sinoquet. *Grammaires à transformations morphiques. Recherche de motif - exacte ou approchée - adaptée aux séquences génétiques : le système GTM*. PhD thesis, IRISA, REnnes, July 1998.

[90] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[91] E. L. L. Sonnhammer, S. R. Eddy, E. Birney, A. Bateman, and R. Durbin. Pfam: multiple sequence alignments and hmm-profiles of protein domains. *Nucleic Acids Research*, 26(1):320–322, 1997.

[92] R. P. Stanley. Differentiably finite power series. *European Journal of Combinatorics*, 1(2):175–188, 1980.

[93] B. Starkie, F. Coste, and M. van Zaanen. The omphalos context-free grammar learning competition. In *Grammatical Inference: Algorithms and Applications*, pages 16–27, 2004.

[94] B. Starkie, F. Coste, and M. van Zaanen. Progressing the state-of-the art in grammatical inference by competition. *AI Communications*, 18(2):93–115, 2005.

[95] K. J. Supowit. Finding a maximum planar subset of a set of nets in a channel. *IEEE Transactions on Computer-Aided Design*, 6(1):93–94, 1987.

[96] Y. Takefuji, D. Ben-Alon, and A.Zaritsky. Neural computing in discovering rna interactions. *Biosystems*, 27(2):85–96, 1992.

[97] J. van der Hoeven. Relax, but don't be too lazy. *Journal of Symbolic Computation*, 34:479–542, 2002.

[98] V.Guignon, C.Chauve, and S.Hamel. An edit distance between rna stem-loops. *SPIRE 2005(LNCS)*, 3772:334–345, 2005.

[99] G. X. Viennot. Heaps of pieces. I. Basic definitions and combinatorial lemmas. In *Combinatoire énumérative (Montreal, Que., 1985/Quebec, Que., 1985)*, volume 1234 of *Lecture Notes in Math.*, pages 321–350. Springer, Berlin, 1986.

[100] M. S. Waterman. Secondary structure of single stranded nucleic acids. *Advances in Mathematics Supplementary Studies*, 1(1):167–212, 1978.

[101] J.D. Watson and F. H. C. Crick. Molecular structures of nucleic acids. *Nature*, 4356:737–738, 1953.

[102] F. Weinberg. Non uniform generation of combinatorial objects. Master's thesis, Universität Kaïserslautern, 2006. Diplomarbeit.

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

[104] H. Yang, F. Jossinet, N. Leontis, L. Chen, J. Westbrook, H.M. Berman, and E. Westhof. Tools for the automatic identification and classification of rna base pairs. *Nucleic Acids Research*, 31(13):4250–4263, 2003.

[105] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[106] M. Zucker. Rna folding by energy minimization.

[107] M. Zucker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406–3415, 2003.

LISTE DES FIGURES

ANNEXE

A. MÉTHODE RÉCURSIVE GÉNÉRALISÉE

Cette application de la méthode récursive à un cas non-algébrique laisse entrevoir la possibilité de généraliser l'approche de Wilf à une approche récursive généralisée. Celle-ci engloberait à la fois l'ensemble des classes décomposables non-étiquetées de Flajolet [42], c'est à dire entre autre les grammaires non-contextuelles, ainsi que les chemins culminants. On présente donc ici une variante de la méthode récursive introduite par Wilf [103] particulièrement adaptée à la génération de marches aléatoires. On remarquera que cette définition ne permet pas de représenter toutes les classes combinatoires prises en charge par le package `CombStruct` de `Maple` [42]. Cependant, elle permet, par l'introduction de la notion de *caractéristique d'une classe d'états*, de clarifier l'évaluation des complexités associées à cette famille d'algorithmes.

A.1 Principe

La méthode récursive, introduite par Wilf [103], repose sur l'idée suivante :

> Si l'on trouve une décomposition non-ambiguë des objets à engendrer, alors on sait compter puis engendrer uniformément ces objets.

On assimile une décomposition pour une classe combinatoire $\mathscr{C}$ à un automate étiqueté sur les arcs et les noeuds.

Définition 28 (Décomposition d'une classe combinatoire) :
Une décomposition pour une classe combinatoire $\mathscr{C}$ est caractérisée par la donnée d'un quintuplet $\mathscr{A} = (Q, q_s, \Phi, \tau, \lambda)$ où :

- *Q est l'ensemble des états de l'automate, aussi appelés* choix locaux.
- *q_s est l'état initial de l'automate.*
- *Φ est un ensemble de fonctions sur la classe combinatoire $\mathscr{C}$.*
- *$\tau : Q \to \oplus \cup (\{\otimes\} \times \Phi) \cup (\{\diamond\} \times \Phi)$ associe à chaque état q un type :*
  - *$\tau(q) = \oplus$: q est un état* disjonctif.
  - *$\tau(q) = (\otimes, f)$: q est un état* conjonctif *muni d'un constructeur f à k arguments, $k = |\lambda(q)|$.*
  - *$\tau(q) = (\diamond, f)$: q est un état* terminal *muni d'un constructeur f sans argument.*

- $\lambda : Q \to (Q \times \Phi)^*$ *définit les transitions de l'automate, c'est à dire un ensemble de couples* (q', f)*, où* f *est un constructeur à un argument.* □

On impose en outre que le graphe dirigé défini implicitement par la fonction λ *soit* acyclique*, c'est à dire :*

$$\nexists (q_1, o_1), \ldots, (q_k, o_k), \quad (q_{i+1}, o_{i+1}) \in \lambda(q_i), \ \textit{tels que } q_1 \rightsquigarrow q_k \to q_1$$

Définition 29 (Ensemble d'objets associés à un état) :
Soit $\mathscr{L}_q$ *l'ensemble des objets dérivables à partir d'un état* q *tel que* $\lambda(q) = ((f_1, q_1), \ldots, (f_k, q_k)$*, alors :*

$$\begin{array}{lll} \tau(q) = \oplus & : & \mathscr{L}_q = \bigcup_{(f,q') \in \lambda(q)} f(\mathscr{L}_{q'}) \\ \tau(q) = (\otimes, f) & : & \mathscr{L}_q = f(f_1(\mathscr{L}_{q_1}), \ldots, f_k(\mathscr{L}_{q_k})) \\ \tau(q) = (\diamond, f) & : & \mathscr{L}_q = f() \end{array}$$

Remarque :
Les fonctions f n-aires associées aux transitions ($n = 1$) et aux états conjonctifs ($n = k$) devront être telles qu'il existe $\mu : \mathbb{N}^k \to \mathbb{N}$ telle que :

$$|f(\mathscr{L}_1, \ldots, \mathscr{L}_k)| = \mu(|\mathscr{L}_1|, \ldots, |\mathscr{L}_k|)$$

où $\mathscr{L}_i$ est l'ensemble d'objets issus de la i-ème transition. Cette propriété garantit l'existence d'une récurrence sur la taille des ensembles considérés, et non sur leur contenu.

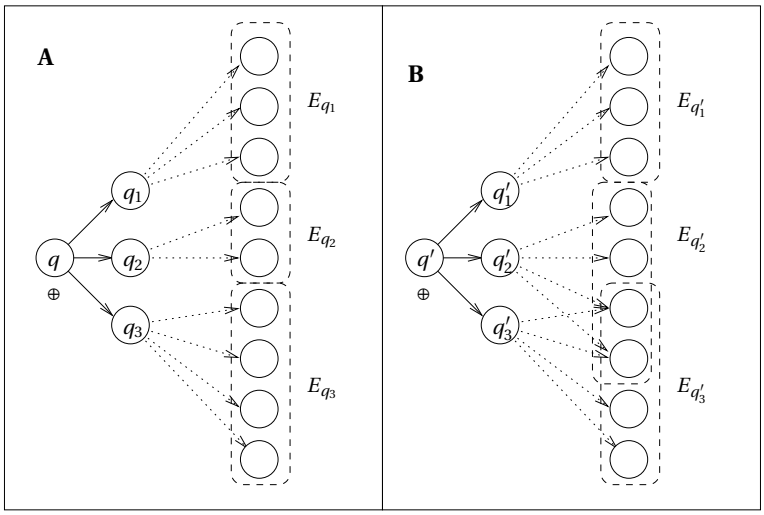


Fig. A.1: Deux décompositions pour une même classe combinatoire. **A** est localement non-ambiguë car $\mathscr{L}_{q_1} \cap \mathscr{L}_{q_2} = \mathscr{L}_{q_2} \cap \mathscr{L}_{q_3} = \mathscr{L}_{q_1} \cap \mathscr{L}_{q_3} = \varnothing$. **B** est ambiguë car $\mathscr{L}_{q'_2} \cap \mathscr{L}_{q'_3} \neq \varnothing$.

Définition 30 (Canonicité d'une décomposition) :
On dit qu'un état ou choix local q, $\lambda(q) = ((f_1, q_1), \ldots, (f_k, q_k))$ est non-ambigu s'il respecte la condition suivante :

$$\begin{array}{lll} \tau(q) = \oplus & : & \forall i \neq j \in [1,k],\ f_i(\mathscr{L}_{q_i}) \cap f_j(\mathscr{L}_{q_j}) = \varnothing \\ \tau(q) = (\otimes, f) & : & \forall \mathbf{X} \neq \mathbf{Y} \in f_1(\mathscr{L}_{q_1}) \times \ldots \times f_k(\mathscr{L}_{q_k}),\ f(\mathbf{X}) \neq f(\mathbf{Y}) \end{array}$$

Un état terminal est toujours non-ambigu.
Une décomposition est non-ambiguë *si tous ses choix locaux sont non-ambigus.*
Une décomposition $\mathscr{A} = (Q, q_s, \Phi, \tau, \lambda)$ d'une classe combinatoire $\mathscr{C}$ est canonique *si elle est non-ambiguë et que $\mathscr{L}_{q_s} = \mathscr{C}$.* □

Remarque :
Les fonctions μ évoquées en remarque à la définition d'une décomposition sont en réalité totalement déterminées dans le cas d'une décomposition canonique :

- État *q disjonctif*, $\tau(q) = \oplus$:
  On rappelle que $\mathscr{L}_q = \cup_{(f,q')\in\lambda q} f(\mathscr{L}_{q'})$ et $c_q = \sum_{(f,q')\in\lambda q} \mu_f(c_{q'})$. Les f sont des fonctions

  $$\Rightarrow |E| \geq |f(E)| \Rightarrow \mu_f \leq \mathbb{I} \Rightarrow c_q \leq \Sigma_{(f,q')\in\lambda q} c_{q'}.$$

  De plus, si $c_q < \Sigma_{(f,q')\in\lambda q} c_{q'}$, alors il existe $(f_1, q_1') \neq (f_2, q_2')$ tels que $f_1(\mathscr{L}_{q_1'}) \cap f_2(\mathscr{L}_{q_2'}) \neq \varnothing$, ce qui contredit l'hypothèse de canonicité (ambiguïté). On a alors $c_q := \sum_{(f,q')\in\lambda(q)} \mu_f(c_{q'}) = \sum_{(f,q')\in\lambda(q)} c_{q'}$, et $(\mu_i \leq \mathbb{I})$, donc :

  $$\mu_{f_i}(n) = n$$

- État *q conjonctif*, $\tau(q) = (\otimes, f)$:
  On rappelle que $\mathscr{L}_q = f(f_1'(\mathscr{L}_{q_1'}), \ldots, f_{|f|}'(\mathscr{L}_{q_{|f|}'}))$, donc $c_q \leq \prod_{(f',q')\in\lambda(q)} c_{q'}$.
  Or $c_q < \prod_{(f',q')\in\lambda(q)} c_{q'}$ implique qu'il existe $(a_1, \cdots, a_{|f|}) \neq (b_1, \cdots, b_{|f|})$ tels que

  $$f(f_1'(a_1), \ldots, f_{|f|}'(a_{|f|})) = f(f_1'(b_1), \ldots, f_{|f|}'(b_{|f|})),$$

  ce qui implique l'ambiguïté de la décomposition. On a donc

  $$c_q := \mu_f\left(\prod_{(f',q')\in\lambda(q)} \mu_{f'}(c_{q'})\right) = \prod_{(f',q')\in\lambda(q)} c_{q'}.$$

  Comme les fonctions μ_i candidates sont inférieures à $\mathbb{I}$, alors il n'y a pas de *compensation* possible à la sublinéarité stricte d'une d'entre elles, donc :

  $$\mu_{f_i'}(n) = n,\ \forall i \in [1, |f|] \quad \text{et} \quad \mu_f(n_1, \ldots, n_{|f|}) = \prod_{i=1}^{|f|} n_i$$

On utilise alors cette décomposition canonique pour dénombrer les cardinalités c_{q_i} des ensembles $\mathscr{L}_{q_i}$ d'objets constructibles à partir de chaque état q_i.

Une fois ces cardinaux calculés, la génération aléatoire uniforme est immédiate. En partant de $q = q_s$ l'état initial de la décomposition de $\mathscr{C}$.
Selon le type de q :

- État *terminal* $\tau(q) = (\diamond, f)$:
  On renvoie $f()$.

- État *conjonctif* $\tau(q) = \oplus$:
  On choisit aléatoirement une des transitions (f_i, q_i) parmi l'ensemble $\lambda(q)$ des transitions selon des probabilités $\mathbb{P}(Y = (f_j, q_j)) = \frac{c_{q_j}}{c_q}$. On engendre alors récursivement un objet ω_i uniformément à partir de q_i et on renvoie $f_i(\omega_i)$.

- État *disjonctif* $\tau(q) = (\otimes, f)$:
  Soit $\lambda(q) = ((f_1, q_1), \ldots, (f_k, q_k))$, on engendre récursivement des objets $\omega_1 \ldots \omega_k$ uniformément à partir des états $q_1 \ldots q_k$, puis on renvoie $f(f_1(\omega_1), \ldots, f_k(\omega_k))$.

La probabilité *a posteriori* $\mathbb{P}(X = \omega | q)$ qu'un objet ω soit engendré à partir d'un état q est donc la suivante :

- État *terminal* $\tau(q) = (\diamond, f)$:
  $\mathbb{P}(X = f() | q) = \frac{1}{c_q}$.

- État *conjonctif* $\tau(q) = \oplus$:
  Soit (f_i, q_i) la transition permettant la création d'un objet ω' tel que $f(\omega') = \omega$, unique par canonicité de la décomposition. On suppose que la génération a bien été uniforme à partir de q_i. On a donc :
  $$\mathbb{P}(X = \omega | q) = \frac{\mathbb{P}(Y = (f_i, q_i))}{c_{q_i}} = \frac{c_{q_i}}{c_q c_{q_i}} = \frac{1}{c_q}$$

- État *disjonctif* $\tau(q) = (\otimes, f)$:
  Soit $\lambda(q) = ((f_1, q_1), \ldots, (f_k, q_k))$. On suppose que la génération des ω_i à partir des q_i est bien uniforme. De plus, dans une décomposition canonique, chaque objet $\omega \in \mathscr{L}_q$ s'obtient à partir de l'application de f à un unique k-uplet $\omega_1 \ldots \omega_k$. On a donc :
  $$\mathbb{P}(X = \omega | q) = \prod_{i=1}^{k} \frac{1}{c_{q_i}} = \frac{1}{c_q}$$

La complexité de cette méthode est dépendante de trois facteurs :

- La longueur des séquences d'états à parcourir avant d'atteindre un état final, celle-ci n'étant pas nécessairement linéaire sur la taille des objets à générer.

- Le nombre de transitions associées à un état disjonctif : Si celui ci est trop important, on devra adopter une stratégie dichotomique, ou bien exploiter des propriétés des c_{q_i}.

- La taille du graphe associé à la décomposition : Elle peut être proportionnelle au nombre d'objets dans la classe, faisant ainsi augmenter déraisonnablement le temps et la mémoire nécessaires à la phase de dénombrement préliminaire. C'est pourquoi il convient de tirer partie autant que possible de symétries dans le graphe représentatif de la décomposition, d'où la notion d'**information caractéristique d'une classe d'états**.

A.1.1 Information caractéristique d'une classe d'états

Il s'agit d'exploiter des symétries dans le graphe représentatif de la décomposition. Pour cela, on va chercher à caractériser en un minimum d'informations les classes d'états sommets d'arbres isomorphes. On *factorisera* ensuite la décomposition de façon à ne faire apparaître qu'une seule copie des graphes isomorphes. On transposera ensuite trivialement les récurrences de dénombrement obtenues sur les états à ces classes.

Exemple :

Considérons les mots de Dyck, exemple historique pour lesquels des algorithmes de génération aléatoires linéaires en temps sont donnés dans [5], [6] ou [78]. Les mots de Dyck $\mathscr{D}$, ou mots *bien*

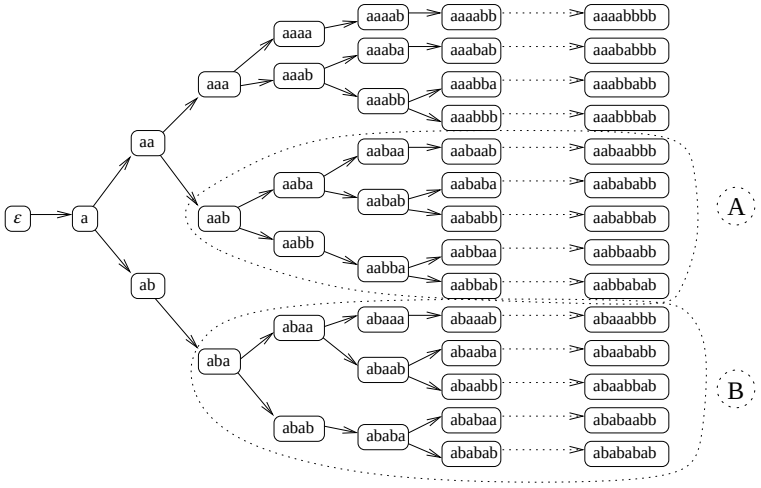


Fig. A.2: Décomposition *marche aléatoire* des mots de Dyck de taille $n = 8$. On remarque que les arbres **A** et **B** issus des états aba et aab sont isomorphes.

parenthèsés, sont les mots $\omega \in \{a,b\}^*$ tels que $|\omega|_a = |\omega|_b$ et $\forall \alpha\beta = \omega$, $|\alpha|_a \geq |\alpha|_b$. On appellera $\mathscr{D}_p$ l'ensemble des préfixes des mots de Dyck, c'est à dire $\mathscr{D}_p = \{\alpha \in \{a,b\}^* \mid \exists \beta \text{ tel que } \alpha\beta \in \mathscr{D}\}$. On adopte une décomposition *itérative* pour ces mots de taille n paire, les assimilant à une *marche aléatoire*, en partant du mot vide ε et à chaque étape, en prolongeant un préfixe α soit par un b si $|\alpha|_a > |\alpha|_b$, soit par un a si $n - |\alpha| < |\alpha|_a - |\alpha|_b$. Cette approche conduit à une décomposition canonique naturelle illustrée par la figure A.2 pour $n = 8$. L'évaluation des c_ω, $\forall \omega \in \mathscr{D}_p$ obéit à la récurrence suivante :

$$c_\omega \quad = \quad \begin{cases} 1 & \text{Si } |\omega| = n \\ \mathbb{1}_{(n-|\omega|<|\omega|_a-|\omega|_b)}\, c_{\omega.a} + \mathbb{1}_{(|\omega|_a>|\omega|_b)}\, c_{\omega b} & \text{Sinon} \end{cases}$$

Dans la figure A.2, on voit apparaître quantité d'arbres isomorphes. Par exemple, $aab \equiv aba$ et $aaab \equiv aaba \equiv abaa$.

En général, on remarquera que la hauteur $h(\omega) = |\omega|_a - |\omega|_b$ et le nombre $n - |\omega|$ de pas à engendrer caractérisent entièrement une classe d'états sommets de sous-arbres isomorphes :

$$\forall \omega, \omega' \in \mathscr{D}_p,\ (|\omega| = |\omega'|) \text{ et } (h(\alpha) = h(\alpha')) \Rightarrow c_\omega = c_{\omega'}$$

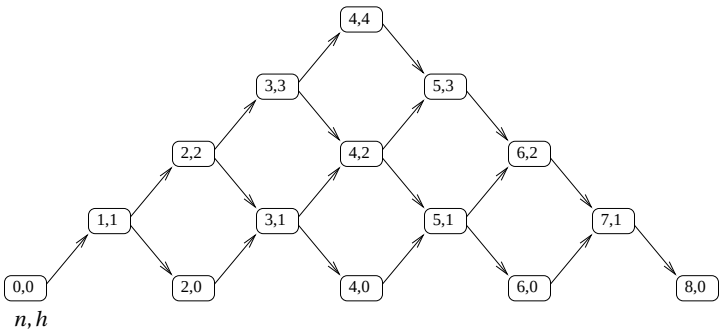


Fig. A.3: Décomposition *factorisée* des mots de Dyck de taille $n = 8$. Les états sont étiquetés par les couples (n,h), longueur/hauteur, des états avant factorisation. Par exemple, l'état $(5,1)$ résulte de la factorisation des états aaabb,aabab,aabba,abaab et ababa.

En effet, on a alors :

$$\begin{aligned}
s \in \mathscr{L}_\omega &\Leftrightarrow \omega.s \in \mathscr{D} \\
&\Rightarrow \begin{cases} h(\omega.s) = 0 \\ |s| + |\omega| = n \\ \forall \alpha.\beta = s,\ h(\omega.\alpha) = h(\omega) + h(\alpha) \geq 0 \end{cases} \\
&\Rightarrow \begin{cases} h(\omega'.s) = h(\omega') + h(s) = h(\omega) + h(s) = 0 \\ |s| + |\omega'| = |s| + |\omega| = n \\ \forall \alpha.\beta = s,\ h(\omega'.\alpha) = h(\omega') + h(\alpha) = h(\omega) + h(\alpha) \geq 0 \end{cases} \\
&\Rightarrow \omega'.s \in \mathscr{D} \text{ car } \omega' \in \mathscr{D}_p \\
&\Leftrightarrow s \in \mathscr{L}_{\omega'} \\
\Rightarrow \mathscr{L}_\omega = \mathscr{L}_{\omega'} &\Rightarrow c_\omega = c_{\omega'}
\end{aligned}$$

On *factorise* donc la décomposition en contractant des états étiquetés par des mots de $\mathscr{D}_p$ ayant mêmes hauteur et longueur. On obtient alors une nouvelle décomposition illustrée par la figure A.3. La récurrence précédente se transpose alors naturellement sur le nouvel espace des états *représentants des classes d'isomorphisme* :

$$c_{k,h} = \begin{cases} 1 & \text{Si } k = 0 \\ \mathbb{1}_{(k<h)} c_{k-1,h+1} + \mathbb{1}_{(h>1)} c_{k-1,h-1} & \text{Sinon} \end{cases}$$

L'algorithme de génération aléatoire est alors le suivant :

1: $\omega \leftarrow \varepsilon$
2: **tant que** $|\omega| < n$ **faire**
3: $r \leftarrow$ RANDOM()
4: **si** $h(\omega) \geq n - |\omega|$ **alors**
5: $\omega \leftarrow \omega.b$
6: **sinon si** $h(\omega) = 0$ **alors**
7: $\omega \leftarrow \omega.a$

8: **sinon si** $r < \frac{c_{n-|\omega|-1,h(|\omega|)+1}}{c_{n-|\omega|,h(|\omega|)}}$ **alors**
9: $\omega \leftarrow \omega.a$
10: **sinon**
11: $\omega \leftarrow \omega.b$
12: **fin si**
13: **fin tant que**

On obtient donc un dénombrement initial en temps[1] et en espace $\Theta(n^2)^*$, car les valeurs k et h prennent des valeurs sur $[0,n]^2$. La génération nécessite n transitions et les alternatives sont au plus au nombre de deux ce qui, au prix d'un léger précalcul supplémentaire[2], indique une complexité en temps $\Theta(n)$.

Remarque : Une astuce combinatoire qui trouve son origine dans [5] et qu'on trouvera notamment dans [84], permet d'éviter le stockage des $c_{k,h}$ et d'optimiser la phase de génération. En effet, il existe une forme *clause* pour les nombres de préfixes/suffixes de Dyck de taille n commençant à hauteur h :

$$c_{k,h} = \frac{h+1}{k+1}\binom{k+1}{\frac{k-h}{2}}$$

Grace à des propriétés sympathiques des binomiaux, on arrive à la forme suivante pour la probabilité d'émission d'une lettre a :

$$\frac{c_{k-1,h+1}}{c_{k,h}} = \frac{h+2}{h+1}\frac{k-h}{2k}$$

Le calcul de probabilité ci dessus mettant en jeu des nombres croissant linéairement sur la taille des mots à engendrer, l'approximation qui consiste à considérer les opérations arithmétiques en temps constant est donc légitime, la génération aléatoire est donc en temps *vraiment* linéaire.

A.1.2 Application aux grammaires non-contextuelles non-ambiguë

Soit G une grammaire non-contextuelle non-ambiguë. Elle admet donc une forme normale de Chomsky $G' = (\Sigma, \mathcal{N}, X, \delta)$, qui se transpose en la décomposition $\mathcal{A}_{G'_n} = (Q_n, q_s, \Phi, \tau, \lambda)$

- $Q_n = (\mathcal{N} \cup \{P_i^S | S \in \mathcal{N},\ S \rightarrow S_1 S_2 \text{ et } i \in [1,n]\}) \times [1,n]$
- $q_s = (X, n)$
- $\Phi = \{\phi_\alpha() = \alpha,\ \forall \alpha \in \Sigma\} \cup \{\phi_\otimes(a,b) = a.b\} \cup \{\phi_\parallel(\omega) = \omega\}$

[1] L'unité de référence étant l'opération arithmétique qui, on va le voir, n'opère pas nécessairement en temps constant c.f. section 3.3.1.

[2] Soit p(resp. q) le nombre de mots de Dyck accessibles en prolongeant par a(resp. b). Alors $\frac{p}{q}$ est rationnel, et admet donc un développement binaire fini ou périodique. L'espérance du nombre de bits aléatoires nécessaires pour *départager* a et b est donc borné par 2. La génération est donc linéaire, quelques soient les tailles des nombres manipulés, à condition de précalculer ces développements binaires périodiques.

- La fonction τ déterminant le *type des états* est telle que :

$$\begin{aligned}\tau((S,i)) &= \begin{cases} \oplus & \text{Si } S \to S_1 \mid S_2 \\ \oplus & \text{Si } S \to S_1 S_2 \\ (\diamond, \phi_\alpha) & \text{Si } S \to \alpha \end{cases} \\ \tau((P_i^S, i)) &= (\otimes, \phi_\otimes)\end{aligned}$$

- Les transitions λ sont quant à elles définies par :

$$\begin{aligned}\lambda((S,i)) &= \begin{cases} \{(\phi_{\mathbb{I}}, (S_1,i)), (\phi_{\mathbb{I}}, (S_2,i))\} & \text{Si } S \to S_1 \mid S_2 \\ \bigcup_{j=1}^{i-1} \{(\phi_{\mathbb{I}}, (P_j^S, i))\} & \text{Si } S \to S_1 S_2 \\ \varnothing & \text{Si } S \to \alpha \end{cases} \\ \lambda((P_j^S, i)) &= \{(\phi_{\mathbb{I}}, (S_1, j)), (\phi_{\mathbb{I}}, (S_2, i-j))\} \text{ où } S \to S_1 S_2\end{aligned}$$

Cette décomposition permet la génération aléatoire uniforme de mots de $\mathscr{L}(G)$ de longueur n en temps $\mathscr{O}(n^2)$ après un précalcul en $\mathscr{O}(n^2)$, dans une implémentation naïve. Cependant on peut adopter, dans la recherche d'une répartition des lettres à engendrer pour un état conjonctif, une stratégie *boustrophédon* similaire à celle de la section 3.3.3 pour obtenir une génération en $\mathscr{O}(n\log(n))$.

A.1.3 Application aux chemins culminants

On considère les chemins culminants $\mathscr{C}^{a,b}$ à pas $+a, -b$, ils sont engendrés dans l'approche récursive généralisée par la décomposition $\mathscr{A}_{\mathscr{C}_n} = (Q'_n, q'_s, \Phi', \tau', \lambda')$ suivante, issue de la récurrence sur les chemins culminants :

- $Q'_n = [0,n] \times [0,an] \times [0,an]$
- $q'_s = (n,0,0)$
- $\Phi' = \{\phi_m(\omega) = m.\omega,\ \phi_{\overline{m}}(\omega) = \overline{m}.\omega,\ \phi_f() = \varepsilon,\ \phi_{\mathbb{I}}(\omega) = \omega\}$
- La fonction τ' déterminant le *type des états* est telle que :

$$\tau'((i,h,y)) = \begin{cases} (\diamond, \phi_f) & \text{Si } i = 0,\ h > y \\ \oplus & \text{Sinon} \end{cases}$$

- Les transitions λ' sont quant à elles définies par :

$$\lambda'((i,h,y)) = \begin{cases} \{(\phi_m(i{-}1, \max(h, y{+}a), y{+}a)), (\phi_{\overline{m}}, (i{-}1, h, y{-}b))\} & \text{Si } i > 1 \text{ et } y > b \\ \{(\phi_m(i{-}1, \max(h, y{+}a), y{+}a))\} & \text{Si } i > 1 \text{ et } y \le b \\ \{(\phi_m(0, \max(h, y{+}a), y{+}a))\} & \text{Si } i = 1 \text{ et } y{+}a > h \\ \varnothing & \text{Sinon} \end{cases}$$

On obtient donc une génération en $\mathcal{O}(n)^*$ après un calcul en $\mathcal{O}(n^3)^*$. Le degré constant des noeuds dans la décomposition, ajouté à la croissance exponentielle des nombres manipulés, suggère le précalcul des probabilités pour les pas montants et descendants, à chaque position. Le surcoût lié au calcul d'une telle division, en $\mathcal{O}(n\log(n)\log\log(n))$ par probabilité si les nombres manipulés croissent exponentiellement, *se fond* dans la complexité de l'addition de ces nombres. Un fois ces probabilités écrites en base 2 et en notation factorisée, (nombres rationnels) l'espérance du nombre de bits à comparer avant de décider en quel état poursuivre la génération est borné par une constante. L'étape de génération est alors en $\Theta(n)$ après un précalcul en $\mathcal{O}(n^{3+\varepsilon})^*$.

A.1.4 Conclusion et perspectives

On a introduit un nouveau formalisme pour exprimer une approche récursive de génération aléatoire. Celle-ci se veut légèrement plus générale que les formalismes existants [41], dans la mesure où ceux-ci voient leur expressivité se borner aux grammaires non-contextuelles. Ici, l'utilisation d'un maximum dans la récurrence, qui ne perturbe absolument pas l'approche choisie, a permis de décrire les chemins culminants, prouvés contextuels au cours du chapitre 9.3.

De plus, il existe de nombreuses classes combinatoires dont la série génératrice est algébrique alors que le langage associé est contextuel. L'aspect contextuel d'un tel langage est alors dû à la non-commutativité du produit de concaténation, et non à des propriétés combinatoires *profondes*. Il n'existe alors pas de grammaire non-contextuelle décrivant le langage. Notre formalisme permet la description de telles classes, et une génération aléatoire efficace.

Ce formalisme pourrait aussi faire l'objet d'une implémentation *générique*, la description de l'ensemble d'états et de la récurrence permettrait alors d'obtenir des générateurs automatiquement. En distinguant une des dimensions de l'espace d'états comme étant la *taille* des séquences, on peut utiliser des optimisations génériques classiques, parmi lesquelles la stratégie boustrophédon (voir section 3.3.3), dans les cas où le nombre de transitions associées à un état est non-borné. Dans le cas d'états ayant un nombre borné de transitions, le précalcul des probabilités associées aux différentes alternatives permettra d'éviter un surcoût lié à la manipulation de grands nombres au cours de la phase de génération.

B. VALIDATION DES GRAMMAIRES PROPOSÉES

Validation de la grammaire inspirée de Waterman pour les structures secondaires d'ARN

S -> a T b S | c S | epsilon

T -> a T b S | c S

Argument 1 : Il s'agit d'engendrer tous les mots de Motzkin sans motif ab, or les Motzkin sont engendrés par S -> a S b S | c S | epsilon. Donc il suffit de créer un nouveau terminal dont le langage soit celui de S moins epsilon et de le substituer à l'occurrence de S séparant une lettre a du b correspondant.

Argument 2 : Calcul des nombres de mots engendrés par la grammaire.

```
> restart;
> SYS1~:= {S=z*T*z*S+z*S+1,T=z*T*z*S+z*S};
```

$$SYS1 := \{S = z^2\,T\,S + z\,S + 1,\ T = z^2\,T\,S + z\,S\}$$

```
> solve(SYS1,{S,T});
```

$$\{S = \text{RootOf}(1 + z^2\,\_Z^2 + (-1 + z - z^2)\,\_Z),\ T = -1 + \text{RootOf}(1 + z^2\,\_Z^2 + (-1 + z - z^2)\,\_Z)\}$$

```
> assign(%);
> allvalues(S);
```

$$\frac{1 - z + z^2 + \sqrt{1 - 2\,z - z^2 - 2\,z^3 + z^4}}{2\,z^2},\ \frac{1 - z + z^2 - \sqrt{1 - 2\,z - z^2 - 2\,z^3 + z^4}}{2\,z^2}$$

```
> series(%[2],z=0,25);
```

$$1 + z + z^2 + 2\,z^3 + 4\,z^4 + 8\,z^5 + 17\,z^6 + 37\,z^7 + 82\,z^8 + 185\,z^9 + 423\,z^{10} + 978\,z^{11} + 2283\,z^{12} + 5373\,z^{13} + 12735\,z^{14} + 30372\,z^{15} + 72832\,z^{16} + 175502\,z^{17} + 424748\,z^{18} + 1032004\,z^{19} + 2516347\,z^{20} + 6155441\,z^{21} + 15101701\,z^{22} + O(z^{23})$$

Un petit coup d'oeil chez Sloane et youpi, il s'agit bien des structures secondaires d'ARN !

(On avait reconnu la s.g. de Watermann, donc le suspense était pas énorme ...)

Grammaire pour les structures secondaires d'ARN admettant des tailles minimales pour les hélices et boucles

Décomposition des mots de Dyck en arbres dont les noeuds sont de degrés non bornés.

Ex : (1 (2)(3)(4)) => 2, 3 et 4 sous-arbres "fils" de 1

Idée de la décomposition : Distinguer les noeuds de degrés 1 (Non Terminal T) de ceux de degrés 0 (NT A) ou >1 (NT B)

En effet, seul les noeuds de degrés 1 peuvent "prolonger" des hélices déjà entamées. Ce sont ces parenthèsages, et uniquement ceux-là, qui ne devront pas être affectés par la taille minimale h des hélices.

On propose la grammaire suivante :

S -> T | A | B

T -> a S b

A -> epsilon

B -> a S b B | a S b a S b

```
> restart;
> SYS2~:= {S=T+A+B,T=z*S*z,A=1,B=z*S*z*B+z^2*S*z*S*z};
```

$$SYS2 := \{S = T + A + B,\ T = z^2\,S,\ A = 1,\ B = z^2\,S\,B + z^4\,S^2\}$$

```
> solve(SYS2,{S,T,U,A,B});
```

$$\{A = 1,\ T = z^2\,\%1,\ S = \%1,\ B = \%1 - 1 - z^2\,\%1,\ U = U\}$$
$$\%1 := \mathrm{RootOf}(-\_Z + 1 + z^2\,\_Z^2)$$

```
> assign(%);
> allvalues(S);
```

$$\frac{1+\sqrt{1-4\,z^2}}{2\,z^2},\ \frac{1-\sqrt{1-4\,z^2}}{2\,z^2}$$

```
> series(%[2],z=0,25);
```

$$1 + z^2 + 2\,z^4 + 5\,z^6 + 14\,z^8 + 42\,z^{10} + 132\,z^{12} + 429\,z^{14} + 1430\,z^{16} + 4862\,z^{18} + 16796\,z^{20} + 58786\,z^{22} + \mathrm{O}(z^{24})$$

=> Catalan !

On est sur la bonne voie, ajoutons les lettres c pour obtenir les mots de Motzkin :

S -> T | A | B

T -> U a S b U

A -> U

B -> U a S b B | U a S b U a S b U

U -> c U | epsilon

```
> restart;
> SYS3~:=
> {S=T+A+B,T=U*z*S*z*U,A=U,B=U*z*S*z*B+U*z*S*z*U*z*S*z*U,U=z*U+1};
```

$$SYS3 := \{S = T + A + B,\ T = U^2 z^2 S,\ A = U,\ B = U z^2 S B + U^3 z^4 S^2,\ U = z U + 1\}$$

```
> solve(SYS3,{S,T,A,B,U});
```

$$\{T = \frac{z^2 \%1}{(-1+z)^2},\ A = -\frac{1}{-1+z},\ B = -\frac{-\%1 + 2\%1\, z + 1 - z}{(-1+z)^2},\ U = -\frac{1}{-1+z},\ S = \%1\}$$

$$\%1 := \mathrm{RootOf}(1 + z^2 \_Z^2 + (-1+z)\_Z)$$

```
> assign(%);
> allvalues(S);
```

$$\frac{1 - z + \sqrt{1 - 2z - 3z^2}}{2z^2},\ \frac{1 - z - \sqrt{1 - 2z - 3z^2}}{2z^2}$$

```
> series(%[2],z=0,25);
```

$$1 + z + 2z^2 + 4z^3 + 9z^4 + 21z^5 + 51z^6 + 127z^7 + 323z^8 + 835z^9 + 2188z^{10} + 5798z^{11} + 15511z^{12} + 41835z^{13} + 113634z^{14} + 310572z^{15} + 853467z^{16} + 2356779z^{17} + 6536382z^{18} + 18199284z^{19} + 50852019z^{20} + 142547559z^{21} + 400763223z^{22} + O(z^{23})$$

=> Motzkin, c'est à dire les structures secondaires d'ARN dans lesquelles la taille minimale pour une boucle est 0 !

Introduisons maintenant les paramètres h et b, tailles minimales respectivement des hélices et boucles :

- Boucles : Facile, il s'agit des noeuds de degrès 0 (NT A) !!!

- Hélices : Plus compliqué, elles sont "prolongées" par certains des noeuds de degrès 1 (NT T), mais pas tous.

Par exemple, dans un noeud unaire du type "a (c a ... b) b", la partie entre parenthèses débute une nouvelle hélice, à cause de la présence de la lettre c, qui correspond dans l'objet biologique correspondant à un renflement qui déstabilise la structure. On "éclate" alors la règle T -> U a S b U en T -> c U a S b | a S b U c | c U a S b U c | a S b.

Seule le dernier terme de la disjonction prolonge une hélice préalablement initiée, toute autre occurence d'un parenthèsage a S b devra être remplacée par a^h S b^h que nous regrouperons en un non-terminal

H-> a^h S b^h .

Enfin, pour être qu'une hélice puisse être prolongée, il faut qu'elle ait été initiée, ce qui n'est pas le cas lors de la toute première réécriture de S en T. On introduit donc un nouvel axiome X, qui "force" l'initiation d'une hélice même dans le cas "unaire pur" a S b => a^h S b^h. On obtient alors la grammaire suivante :

X -> U H U | A | B

S -> T | A | B

T -> c U H | H U c | c U H U c | a S b

A -> c^b U

B -> U H B | U H U H U

U -> c U | epsilon

H -> a^h S b^h

```
> restart;h:=1;b:=1;
```

$$h := 1$$
$$b := 1$$

```
> SYS4~:= {
> X=U*H*U+A+B,
> S=T+A+B,
> T=z*U*H*z*U+H*z*U+z*U*H+z*S*z,
> A=z^b*U,
> B=U*H*B+U*H*U*H*U,
> U=z*U+1,
> H=z^h*S*z^h};
```

$$SYS4 := \{T = z^2 U^2 H + 2 H z U + z^2 S,\ B = U H B + U^3 H^2,\ U = z U + 1,\ H = z^2 S,\ A = z U,$$
$$X = U^2 H + A + B,\ S = T + A + B\}$$

```
> solve(SYS4,{S,T,A,B,U,H,X});
```

$$\{S = \%1,\ U = -\frac{1}{-1+z},\ B = \frac{\%1 - 2\,\%1\,z - z + z^2}{(-1+z)^2},\ X = \%1,\ H = z^2\,\%1,\ A = -\frac{z}{-1+z},$$
$$T = \frac{z^2\,\%1}{(-1+z)^2}\}$$
$$\%1 := \mathrm{RootOf}(z^2\_Z^2 + (-1 + z^2 + z)\_Z + z)$$

```
> assign(%);
> allvalues(X);
```

$$\frac{1 - z^2 - z + \sqrt{1 - z^2 - 2z + z^4 - 2z^3}}{2z^2},\ \frac{1 - z^2 - z - \sqrt{1 - z^2 - 2z + z^4 - 2z^3}}{2z^2}$$

```
> series(%[2],z=0,26);
```

$$z + z^2 + 2\,z^3 + 4\,z^4 + 8\,z^5 + 17\,z^6 + 37\,z^7 + 82\,z^8 + 185\,z^9 + 423\,z^{10} + 978\,z^{11} + 2283\,z^{12} + 5373\,z^{13} + 12735\,z^{14} + 30372\,z^{15} + 72832\,z^{16} + 175502\,z^{17} + 424748\,z^{18} + 1032004\,z^{19} + 2516347\,z^{20} + 6155441\,z^{21} + 15101701\,z^{22} + 37150472\,z^{23} + \mathrm{O}(z^{24})$$

Or, d'après Sloane : Les structures secondaires sont en nombre 1, 1, 1, 2, 4, 8, 17, 37, 82, 185, 423, 978, 2283, 5373, 12735, 30372, 72832, 175502, 424748, 1032004, 2516347, 6155441, 15101701, 37150472, 91618049, 226460893

Et puis la s.g. est correcte pour h=1, b=1, donc il n'y a plus qu'à s'assurer que toute hélice est débuté par a^h S b^h pour conclure sur la correction de la grammaire.