

Structures combinatoires pour l'analyse de génomes

Mémoire présenté le 7 décembre 2004

pour obtenir

l'Habilitation à diriger les recherches

en Sciences mathématiques (spécialité informatique)

par

Hélène Touzet

Composition du jury

Rapporteurs : Alain Denise, professeur, Université Paris-Sud
Jacques van Helden, professeur, Université Libre de Bruxelles
Marie-France Sagot, DR INRIA, projet Helix et UMR CNRS 5558

Examineurs : Maxime Crochemore, professeur, Université Marne-la-Vallée
Serge Dulucq, professeur, Université de bordeaux I
Christine Gaspin, DR INRA, INRA Toulouse
Sophie Tison, professeur, Université des Sciences et Technologies de Lille
Bernard Vandenbunder, DR CNRS, Institut de Biologie de Lille

Directeur : Max Dauchet, professeur, Université des Sciences et Technologies de Lille

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

Laboratoire d'Informatique Fondamentale de Lille — UMR 8022

U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX

Table des matières

Avant-propos	1
1 Expressivité des systèmes de réécriture	3
1.1 Introduction	3
1.2 Terminaison par interprétation polynomiale	5
1.2.1 Définition	5
1.2.2 Complexité des fonctions	6
1.2.3 Cas non-déterministe	7
1.3 Expressivité de la terminaison totale	9
1.3.1 Le plongement homéomorphique	9
1.3.2 La réécriture de mots	9
1.3.3 La réécriture de termes	11
2 La bio-informatique de l'ARN	15
2.1 L'ARN dans la cellule	15
2.2 Structure de l'ARN	17
2.3 Le statut privilégié de la structure secondaire	18
2.3.1 L'analyse informatique des ARN	18
2.3.2 Comment évoluent les ARN	20
2.3.3 Structure secondaire vs structure tertiaire	21
2.3.4 Représentation de la structure secondaire	21
3 Algorithmes pour la comparaison de molécules d'ARN	27
3.1 Arbres et forêts	27
3.1.1 Notations	27
3.1.2 Distance d'édition	28

3.1.3	Sous-forêts	28
3.2	Algorithmes pour la distance d'édition entre arbres	29
3.2.1	Le graphe d'édition	29
3.2.2	Algorithme de Zhang-Shasha	30
3.2.3	Algorithme de Klein	31
3.2.4	Analyse des stratégies de décomposition	33
3.3	Alignement d'arbres	36
3.3.1	Alignement versus distance	36
3.3.2	Algorithme de Jiang pour l'alignement	38
3.4	Prise en compte des gaps	39
3.4.1	Le problème de la définition	39
3.4.2	Gaps affines	40
3.4.3	Gaps convexes	41
3.4.4	Gaps avec sous-arbres complets	41
3.5	Comparaison locale	42
3.5.1	Rechercher les sous-arbres complets optimaux	43
3.5.2	Rechercher les forêts closes optimales	43
3.5.3	Rechercher les sous-arbres optimaux	43
3.6	Intégration de la structure primaire et de la structure tertiaire	45
3.7	Perspectives	47
4	Prédiction de structures secondaires	49
4.1	Position du problème	49
4.2	Carnac	51
4.2.1	Prédiction deux à deux	51
4.2.2	Combinaison des repliements	53
4.3	Résultats expérimentaux	55
4.4	Perspectives	62
5	Analyse des régions régulatrices	65
5.1	La régulation transcriptionnelle	65
5.2	Le modèle des matrices	66
5.3	Localisation à grande échelle des sites de fixation	67

	iii
5.3.1 Groupement de matrices	69
5.3.2 Indexation des matrices	70
5.4 Régions exceptionnelles	71
5.4.1 Futilité des prédictions	71
5.4.2 Sur-représentation locale	71
5.4.3 Les facteurs Rel/NF- κ B et leurs gènes cibles	75
5.5 Perspectives	78
Bibliographie	79
Annexe administrative	87

Avant-propos

Ce mémoire est une synthèse des travaux de recherches menés depuis mon arrivée au LIFL en 1998. Il s'organise en trois thèmes, décrits dans l'ordre chronologique.

La première partie, constituée du chapitre un, traite de l'expressivité des systèmes de réécriture. Ces travaux sont dans la continuité de ma thèse soutenue en septembre 1997 au LORIA sous la direction d'Adam Cichon. Ce sont mes racines. Je présente ici deux nouveaux résultats : la classe de complexité des fonctions qui admettent des interprétations polynomiales et la borne supérieure de la terminaison totale.

Ma nomination au LIFL s'est ensuite accompagnée d'une conversion thématique progressive vers la bio-informatique, et c'est le sujet des chapitres suivants. Les chapitres deux, trois et quatre sont consacrés à l'algorithmique des structures d'ARN. Nous y abordons le problème de la comparaison d'ARN à travers la comparaison de structures secondaires. D'un point de vue combinatoire, cette question se décline en terme de distance entre arbres. La seconde direction de recherche est l'inférence de structures pour des familles d'ARN homologues, qui a abouti à la réalisation du logiciel Carnac dans le cadre de la thèse d'Olivier Perriquet, soutenue en 2003.

L'objet du dernier chapitre est l'analyse bio-informatique des régions régulatrices. Ce thème rejoint la dynamique scientifique créée autour de la modélisation des interactions moléculaires avec déploiement de l'IRI sur Lille. Nous attaquons ce problème par le biais de l'analyse à grande échelle des sites de fixation de facteurs de transcription. Ce sont les sujets des thèses de Matthieu Defrance, débutée en 2003, et d'Aude Liefoghe, qui commence cet automne.

Cette énumération peut avoir des allures d'inventaire à la Prévert. Il y a toutefois une logique sous-jacente. Concernant les aspects bio-informatiques, les études des structures d'ARN et des signaux de régulation s'inscrivent dans une même perspective d'analyse des génomes. Elles participent aux efforts d'annotation des « zones d'ombre », les régions non codantes conservées révélées par les programmes de séquençage. De manière plus générale, tous ces travaux ont été nourris par le même plaisir de faire de la recherche avec des chercheurs en mathématique, en informatique et maintenant en biologie.

Le document a été rédigé en essayant de se dégager des contingences techniques, pour mettre en valeur l'esprit plutôt que la lettre. Les démonstrations sont donc omises et pourront être trouvées dans les différentes publications associées à ce travail [10, 26, 30, 107, 133, 134, 135].

Chapitre 1

Expressivité des systèmes de réécriture

Ce premier chapitre est consacré à l'analyse de la complexité des calculs dans le cadre de la théorie de la réécriture. Comme pour tous les modèles de calcul, deux questions naturelles se posent : le calcul termine-t-il et, si oui, en combien de temps. Nous développons cet aspect à travers deux problèmes : la classe de complexité des fonctions qui admettent des interprétations polynomiales en section 1.2 et la borne supérieure de la terminaison totale en section 1.3.

1.1 Introduction

La réécriture est un modèle de calcul à base de règles équationnelles du premier ordre. La légèreté de sa syntaxe, son évidente analogie avec la programmation fonctionnelle en font un cadre approprié pour étudier les propriétés des programmes.

Formellement, un *système de réécriture* \mathcal{R} est défini par

- un ensemble de symboles de fonctions \mathcal{F} ,
- un ensemble de variables \mathcal{X} ,
- un ensemble de règles $l \rightarrow r$ construites sur l'algèbre des termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

La *relation de réécriture* définie par \mathcal{R} est la plus petite relation monotone, stable et transitive induite par les règles de \mathcal{R} . Un terme u se réécrit en un terme v en une étape de calcul s'il existe un terme $t(x)$ admettant exactement une seule occurrence de x , une substitution $\sigma : \mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ et une règle $l \rightarrow r$ de \mathcal{R} satisfaisant $u = t(l\sigma)$ et $v = t(r\sigma)$.

La terminaison des systèmes de réécriture est indécidable en général et reste indécidable dans de nombreux cas particuliers [64, 14, 25, 45]. Pour remédier à ce problème, la démonstration automatique a développé des techniques de preuve de terminaison destinées à l'analyse des systèmes de réécriture utilisés dans la spécification de programmes. Il s'agit par exemple des *preuves par interprétation*, basées sur une analyse sémantique du système de réécriture, comme les interprétations polynomiales ou plus généralement à base de fonctions croissantes sur les entiers. D'autres approches encore proposent des critères syntaxiques simples pour établir que la relation de réécriture est compatible avec un ordre bien fondé de l'algèbre de termes. C'est le cas des *ordres récursifs sur les chemins* (MPO, Multiset Path Ordering [28], et LPO, Lexicographic Path Ordering [73]) et de l'*ordre de Knuth-Bendix* KBO [81]. Toutes ces méthodes établissent en fait la terminaison totale (qui est également indécidable [154]).

Définition 1 (Terminaison totale). *Soit $\mathcal{T}(\mathcal{F}, \mathcal{X})$ une algèbre de termes. Un ordre de terminaison totale est un ordre bien-fondé total sur $\mathcal{T}(\mathcal{F}, \mathcal{X})$ et monotone sur $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Un système*

de réécriture \mathcal{R} défini sur $\mathcal{T}(\mathcal{F}, \mathcal{X})$ termine totalement s'il existe un ordre de terminaison totale compatible avec la relation de réécriture associée.

Un problème corollaire à la terminaison est l'expressivité des ordres de terminaison. Une fois la terminaison d'un système de réécriture établie, quelle information peut-on avoir concernant la complexité du calcul associé? On définit les fonctions $dl_{\mathcal{R}}$ et $Dl_{\mathcal{R}}$ par :

$$\begin{aligned} dl_{\mathcal{R}} : \quad \mathcal{T}(\mathcal{F}, \mathcal{X}) &\rightarrow \mathbb{N} \\ t &\mapsto \max\{dl_{\mathcal{R}}(u) + 1 ; t \rightarrow_{\mathcal{R}} u\} \\ \\ Dl_{\mathcal{R}} : \quad \mathbb{N} &\rightarrow \mathbb{N} \\ m &\mapsto \max\{n \in \mathbb{N} ; \exists t \in \mathcal{T}(\mathcal{F}, \mathcal{X}), dl_{\mathcal{R}}(t) = n \wedge |t| \leq m\} \end{aligned}$$

$Dl_{\mathcal{R}}$ est appelée la *complexité dérivationnelle* ou la *longueur de dérivation* de \mathcal{R} . Une analyse quantitative de la terminaison présente plusieurs intérêts. Cela permet d'une part de déterminer l'expressivité des techniques de preuves utilisées, de les classifier et de jauger leur limite. Le second aspect est d'ordre pratique : la complexité dérivationnelle donne une information sur la longueur du calcul associé au système de réécriture, et donc sur la complexité d'algorithmes de résolution, de réduction reposant sur ce système. Enfin, la complexité dérivationnelle permet d'analyser les fonctions définies directement par les systèmes de constructeurs.

Définition 2 (Système de constructeurs). *Un système de constructeurs est un système de réécriture pour lequel la signature peut être partitionnée en deux ensembles : les symboles définis \mathcal{D} et les symboles de constructeurs \mathcal{C} . Tous les membres gauches de règles de réécriture sont de la forme $f(t_1, \dots, t_n)$ où f est un symbole défini et t_1, \dots, t_n sont des termes de $\mathcal{T}(\mathcal{C}, \mathcal{X})$. Les termes clos en forme normale appartiennent à $\mathcal{T}(\mathcal{C})$.*

Définition 3 (Représentation d'une fonction). *Soit f , une fonction de $A \rightarrow B$. f est représentée ou calculée par un système de constructeurs \mathcal{R} s'il existe un codage $\ulcorner \cdot \urcorner : A \cup B \rightarrow \mathcal{T}(\mathcal{C})$ et un symbole défini $\lceil f \rceil$ de même arité que f , tels que pour tout u de A , $\lceil f \rceil(\ulcorner u \urcorner)$ admette une unique forme normale, égale à $\ulcorner f(u) \urcorner$.*

Dans ce cadre, la longueur de dérivation donne une information naturelle sur la classe de complexité de la fonction représentée. C'est une conséquence des deux théorèmes suivants.

Théorème 1 (Kleene [76]). *Soit $f : \mathbb{N}^p \rightarrow \mathbb{N}$ une fonction calculable par une machine de Turing \mathcal{M} et soit $T_{\mathcal{M}} : \mathbb{N}^p \rightarrow \mathbb{N}$, tel que $T_{\mathcal{M}}(x_1, \dots, x_p)$ est le temps de calcul de \mathcal{M} sur l'entrée (x_1, \dots, x_p) . Alors il existe des fonctions récursives primitives $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ et $\text{Sit} : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ telles que*

$$\forall (x_1, \dots, x_p) \in \mathbb{N}^p \quad f(x_1, \dots, x_p) = \alpha(\text{Sit}(T_{\mathcal{M}}(x_1, \dots, x_p), x_1, \dots, x_p)).$$

Théorème 2 (Handley - Wainer [51]). *Soit $f : \mathbb{N}^p \rightarrow \mathbb{N}$, une fonction représentée par un système de réécriture \mathcal{R} . Alors il existe une machine de Turing \mathcal{M} calculant f et une fonction élémentaire $h : \mathbb{N} \rightarrow \mathbb{N}$, telles que*

$$\forall x_1, \dots, x_p \in \mathbb{N} \quad dl_{\mathcal{R}}(f_{\mathcal{R}}(\ulcorner x_1 \urcorner, \dots, \ulcorner x_p \urcorner)) \leq h(T_{\mathcal{M}}(x_1, \dots, x_p)),$$

où $f_{\mathcal{R}}$ est la fonction qui représente f dans \mathcal{R} et où la fonction $T_{\mathcal{M}} : \mathbb{N}^p \rightarrow \mathbb{N}$ est définie comme dans le théorème 1.

Corollaire 1. *Soit f une fonction sur les entiers.*

1. Si f est représentée par un système de réécriture dont la longueur de dérivation est récursive primitive, alors f est elle-même récursive primitive.
2. Si f est représentée par un système de réécriture dont la longueur de dérivation est récursive multiple, alors f est elle-même récursive multiple.

Les classes de complexité des longueurs de dérivation autorisées pour les interprétations polynomiales et les trois ordres MPO, LPO et KBO ont été caractérisées.

Théorème 3.

- Les systèmes qui admettent une interprétation polynomiale ont une longueur de dérivation doublement exponentielle (Hofbauer et Lauteman [62]).
- Les systèmes de réécriture qui réduisent sous MPO ont une longueur de dérivation primitive récursive (Hofbauer [60], Buchholz [13]).
- Les systèmes de réécriture qui réduisent sous KBO ont une longueur de dérivation 2-récursive (Lepper [85], Hofbauer [61]).
- Les systèmes de réécriture qui réduisent sous LPO ont une longueur de dérivation multiple récursive (Weiermann [146], Buchholz [13]).

Dans chacun des quatre cas, la borne est optimale.

Avec le corollaire 1, il s'ensuit que les fonctions qui terminent avec MPO sont exactement les fonctions récursives primitives, et que les fonctions qui terminent avec LPO sont exactement les fonctions multiples récursives.

Ces résultats peuvent s'analyser à au moins deux niveaux. Le premier niveau est celui de l'utilisateur, du programmeur. Celui-ci ne peut qu'exprimer sa frustration face au caractère non opérationnel des classes de complexité impliquées. Dans la hiérarchie de Grzegorzcyk, qui fournit une échelle de la complexité des fonctions, les fonctions exponentielles apparaissent à l'indice 3, les fonctions primitives récursives à l'indice ω et les fonctions multiples récursives à l'indice ω^ω . En pratique, un calcul de même longueur que la fonction d'Ackermann, d'indice ω , ou un calcul qui ne termine pas font peu de différence. Il faudrait pouvoir dégager des critères simples sur les ordres de terminaison qui garantissent que les fonctions soient calculables de manière effective. Nous abordons cette question en section 1.2, avec une nouvelle description de la classe PTIME dans le cadre des interprétations polynomiales. Ces travaux s'inscrivent dans un projet plus général de confronter les classes de complexité habituelles aux fonctions définissables par un système de réécriture, avec l'espoir d'obtenir de nouvelles caractérisations simples.

Le second point de vue est purement logique. Il s'agit de la limite théorique des méthodes de preuve de terminaison qui utilisent les ordres de terminaison totale. L'ordre LPO, qui affiche les plus longues dérivations, est-il le plus complexe possible? Cette question pose le problème de l'expressivité du théorème de Kruskal, et de la relation entre le type d'ordre d'un ordre de terminaison totale et sa complexité dérivationnelle. C'est le sujet de la section 1.3.

1.2 Terminaison par interprétation polynomiale

1.2.1 Définition

Une preuve de terminaison par interprétation consiste à associer à chaque symbole de fonction g de \mathcal{F} une fonction sur les entiers positifs $[g]$ compatible avec le système de réécriture. Plus précisément, $[g]$ doit satisfaire

- $[g]$ admet un nombre de variables égal à l'arité de g ,
- $[g]$ est monotone, c'est-à-dire $n < m \Rightarrow [g](\dots, n, \dots) < [g](\dots, m, \dots)$
- Si g est un symbole de constante, alors $[g] > 0$. Dans les autres cas, $[g](n_1, \dots, n_k) > n_i$, pour $i \in [1, k]$.

L'interprétation $[\]$ est étendue aux termes clos de manière canonique :

$$[g(t_1, \dots, t_n)] = [g]([t_1], \dots, [t_n]).$$

La compatibilité avec le système de réécriture s'écrit : pour toute règle $l \rightarrow r$ de \mathcal{R} , on a $[l] > [r]$ sur le domaine des interprétations des termes clos. Les interprétations les plus couramment utilisées dans les systèmes de preuve automatique de terminaison sont les interprétations polynomiales sur les entiers ([126], Polo [46], Torpa [155],...). Assigner une interprétation polynomiale assure la terminaison totale.

Exemple 1. *Ce système de réécriture permet de définir la fonction carré, via la définition de l'addition et de la multiplication. Les constructeurs sont 0 et s.*

$$\left\{ \begin{array}{l} \text{Add}(0, y) \rightarrow y \\ \text{Add}(sx, y) \rightarrow s(\text{Add}(x, y)) \\ \text{Mul}(0, y) \rightarrow 0 \\ \text{Mul}(sx, y) \rightarrow \text{Add}(y, \text{Mul}(x, y)) \\ \text{Sq}(x) \rightarrow \text{Mul}(x, x) \end{array} \right.$$

Il admet comme interprétation

$$\begin{aligned} [0] &= 2 \\ [s](x) &= x + 1 \\ [\text{Add}](x, y) &= 2x + y \\ [\text{Mul}](x, y) &= 3xy \\ [\text{Sq}](x) &= 3x^2 + 1 \end{aligned}$$

1.2.2 Complexité des fonctions

Le théorème 3 indique que la terminaison par interprétation polynomiale n'assure pas la terminaison en temps polynomial. On peut obtenir davantage d'informations en examinant le contenu intentionnel des interprétations des symboles de constructeurs [20, 62]. L'exemple 2 motive cette démarche.

Exemple 2. *On étend le système de réécriture de l'exemple 1 pour définir la fonction factorielle*

$$\left\{ \begin{array}{l} \text{Fact}(0) \rightarrow s(0) \\ \text{Fact}(s(x)) \rightarrow \text{Mul}(s(x), \text{Fact}(x)) \end{array} \right.$$

Il est facile de vérifier, en raisonnant sur le degré, qu'il n'existe pas d'interprétation polynomiale pour Fact compatible avec les interprétations précédentes de s, Add et Mul. Il est toutefois possible de s'affranchir de cette limite par le truchement d'une astuce syntaxique. On introduit un nouveau symbole de constructeur q, et une fonction de traduction Trad pour gérer les deux types d'entiers.

$$\left\{ \begin{array}{l} \text{Fact}(0) \rightarrow s(0) \\ \text{Tr}(q(x)) \rightarrow s(\text{Tr}(x)) \\ \text{Tr}(0) \rightarrow 0 \\ \text{Fact}(q(x)) \rightarrow \text{Mul}(s(\text{Tr}(x)), \text{Fact}(x)) \end{array} \right.$$

La forme normale de $\text{Fact}(q^n(0))$ est bien $s^{n!}(0)$. Ce système alternatif admet une interprétation polynomiale :

$$\begin{aligned} [q](x) &= 3(x+2)^2 \\ [\text{Tr}](x) &= x+1 \\ [\text{Fact}](x) &= x+2 \end{aligned}$$

Cet exemple met en évidence le rôle sensible des constructeurs et des interprétations des constructeurs. La contribution des constructeurs peut être capturée par la définition de trois classes, suivant le type du polynôme servant d'interprétation.

Classe 0 : polynômes de la forme $P(X_1, \dots, X_n) = X_1 + \dots + X_n + c$, avec $c > 0$,

Classe 1 : polynômes de la forme $P(X_1, \dots, X_n) = a_1X_1 + \dots + a_nX_n + b$ avec $\max\{a_1, \dots, a_n\} > 1$ et $b \geq 0$,

Classe 2 : polynômes de la forme $P(X_1, \dots, X_n) = X_1^{\beta_1} \times \dots \times X_n^{\beta_n} + R(X_1, \dots, X_n)$ où $\max\{\beta_1, \dots, \beta_n\} > 1$ et R est un polynôme quelconque.

Cette classification permet de distinguer trois familles de systèmes de réécriture.

Définition 4. *Pour tout i de $\{0, 1, 2\}$, un système de réécriture est $\Pi(i)$ s'il admet une interprétation polynomiale pour laquelle les constructeurs sont interprétés par un polynôme de la classe j , avec $j \leq i$. Une fonction est $\Pi(i)$ -calculable s'il existe un système de réécriture $\Pi(i)$ qui permet de la calculer.*

L'exemple 1 montre que l'addition et la multiplication sont des fonctions $\Pi(0)$ -calculables, et l'exemple 2 que la fonction factorielle est $\Pi(2)$ -calculable.

Théorème 4 (Bonfante, Cichon, Marion et Touzet [10]).

1. *Les fonctions $\Pi(0)$ -calculables sont exactement les fonctions de la classe PTIME des fonctions calculables en temps polynomial.*
2. *Les fonctions $\Pi(1)$ -calculables sont exactement les fonctions calculables en temps exponentiel.*
3. *Les fonctions $\Pi(2)$ -calculables sont exactement les fonctions calculables en temps doublement exponentiel.*

1.2.3 Cas non-déterministe

La réécriture fournit également un modèle de calcul non-déterministe, avec les systèmes de réécriture non confluents. *Modèle de calcul* s'entend ici au sens de [82]. On munit l'ensemble des termes clos de $\mathcal{T}(\mathcal{C})$ d'une relation d'ordre, et une fonction f est calculable par un système de réécriture non-confluent si $\ulcorner f(u) \urcorner = \max\{v \in \mathcal{T}(\mathcal{C}) \mid \ulcorner f \urcorner(\ulcorner u \urcorner) \rightarrow v\}$. Cela conduit à une classification analogue à celle de la définition 4 : une fonction est $\Delta(i)$ -calculable s'il existe un système de réécriture (éventuellement non confluent) avec une interprétation polynomiale de classe $\leq i$ qui la calcule au sens de la définition précédente.

Théorème 5 (Bonfante, Cichon, Marion et Touzet [10]).

1. *Les fonctions $\Delta(0)$ -calculables sont exactement les fonctions de la classe NPTIME des fonctions calculables en temps polynomial non déterministe.*
2. *Les fonctions $\Delta(1)$ -calculables sont exactement les fonctions calculables en temps exponentiel non-déterministe.*

3. Les fonctions $\Delta(2)$ -calculables sont exactement les fonctions calculables en temps doublement exponentiel non-déterministe.

Exemple 3. On peut calculer une clique maximale dans un graphe par un système de réécriture non-confluent. Un graphe est décrit par une liste de nœuds, V , et une liste d'arcs, E , un arc étant une liste de deux nœuds. Les symboles de fonctions `is_edge`, `is_connected` et `is_clique` sont des prédicats : `is_edge(u, v, E)` teste si l'arc (u, v) appartient à E , `is_connected(u, V, E)` teste si le nœud u est connecté à tous les nœuds de V par les arcs de E et `is_clique(V, E)` teste si les nœuds de V sont totalement connectés par E .

Pour les constructeurs, `nil` et `*` permettent de construire des listes, et donc des graphes, sous forme de listes de nœuds et de listes d'arcs. On introduit deux symboles de constante supplémentaires `true` et `false` pour représenter les booléens. L'ordre sous-jacent est `false < true`, et les listes sont ordonnées suivant leur taille effective.

```

and(true, true)  → true
and(x, y)       → false

if_then_else(true, x, y) → x
if_then_else(false, x, y) → y

is_edge(u, v, nil) → false
is_edge(u, v, (u * v) * E) → true
is_edge(u, v, (u' * v') * E) → is_edge(u', v', E)

is_connected(u, nil, E) → false
is_connected(u, v * V, E) → and(is_edge(u, v, E), is_connected(u, V, E))

is_clique(nil, E) → true
is_clique(u * K, E) → and(is_connected(u, K, E), is_clique(K, E))

build_clique(nil, K, E) → if_then_else(is_clique(K, E), K, nil)
build_clique(u * V, K, E) → build_clique(V, u * K, E)
build_clique(u * V, K, E) → build_clique(V, K, E)

```

La fonction `build_clique` est la fonction finale, qui construit les cliques. L'ensemble des formes normales de `build_clique(V, nil, E)` est exactement l'ensemble des cliques du graphe décrit par V et E . Le système admet une interprétation polynomiale, avec pour les symboles de constructeur

$$\begin{array}{ll}
[\text{true}] &= 1 & [\text{nil}] &= 1 \\
[\text{false}] &= 1 & [x * y] &= x + y + 1
\end{array}$$

et pour les symboles définis

$$\begin{array}{ll}
[\text{and}](x, y) &= x + y \\
[\text{if_then_else}](x, y, z) &= x + y + z \\
[\text{is_edge}](x, y, z) &= x + y + z \\
[\text{is_connected}](x, y, z) &= (x + z)y \\
[\text{is_clique}](x, y) &= x^2y \\
[\text{build_clique}](x, y, z) &= (2x + y)^2z
\end{array}$$

Cela montre que la fonction `build_clique` est $\Delta(0)$ -calculable : le problème est dans classe NP.

1.3 Expressivité de la terminaison totale

Nous nous intéressons maintenant à la borne supérieure de l'expressivité des ordres de terminaison totale, à leur limite théorique. Le théorème 3 montre que les méthodes connues autorisent au mieux des longueurs de dérivation multiples récursives avec LPO. Est-il possible de définir des systèmes de réécriture de complexité supérieure dans le cadre de la terminaison totale? C'est le sujet de cette section.

Les ordres de terminaison totale sont des ordres bien fondés totaux, c'est-à-dire des bons ordres. Il existe un cadre logique privilégié pour l'étude des bons ordres et de leur complexité : les ordinaux, qui sont des représentants canoniques des bons ordres. Nous ne rappelons pas ici les fondements de la théorie des ordinaux et les différents systèmes de notation. Des références classiques sont [108, 123] ou le récent et joliment nommé *Why ordinals are good for you* [87].

Indépendamment des systèmes de réécriture et de leur longueur de dérivation, la complexité d'un ordre de terminaison totale peut être mesurée par son type d'ordre, c'est-à-dire l'ordinal auquel il est isomorphe. Nous commençons par replacer les ordres de terminaison totale dans le contexte du théorème de Kruskal, avec le plongement homéomorphique.

1.3.1 Le plongement homéomorphique

Définition 5 (Plongement homéomorphique). *Soit $(\mathcal{T}(\mathcal{F}, \mathcal{X}))$ une algèbre de termes. Le plongement homéomorphique $(\mathcal{T}(\mathcal{F}, \mathcal{X}), \trianglelefteq)$ est le plus petit ordre satisfaisant les propriétés suivantes :*

- monotonie : si $u \trianglelefteq v$, alors $f(u) \trianglelefteq f(v)$,
- sous-terme : $t \trianglelefteq f(\dots, t, \dots)$,
- élimination : $f(\dots) \trianglelefteq f(\dots, t, \dots)$.

Le plongement homéomorphique induit une notion de simplicité syntaxique : un terme u est inférieur à un second terme v si u peut être déduit de v en effaçant des nœuds de v .

Théorème 6 (Kruskal [83]). *Soit $\mathcal{T}(\mathcal{F}, \mathcal{X})$ une algèbre de termes engendrée par une signature finie. Le plongement $(\mathcal{T}(\mathcal{F}, \mathcal{X}), \trianglelefteq)$ est un bel-ordre-partiel.*

Un bel-ordre-partiel est un ordre bien-fondé sans anti-chaînes infinies. En d'autres termes, tout ordre qui étend le plongement de Kruskal \trianglelefteq est un ordre bien-fondé. Le théorème de Kruskal fournit ainsi un critère pour la définition d'ordres de terminaison, qui est en fait utilisé de manière dissimulée à chaque fois qu'on prouve la terminaison totale. En effet, sur une signature finie avec des symboles d'arité fixe, les ordres de terminaison totale sont des ordres partiels compatibles avec le plongement homéomorphique [36].

Cela implique que le type d'ordre d'un ordre de terminaison totale est borné par le type d'ordre maximal du plongement homéomorphique. Quelle est la conséquence de cette remarque pour la complexité dérivationnelle? Il faut regarder le cas des systèmes de réécriture de mots, et le cas général des systèmes de réécriture de termes.

1.3.2 La réécriture de mots

Le cas particulier du théorème de Kruskal sur les mots est connu comme le *lemme de Higman*, formulé à partir de l'*ordre de division*.

Définition 6 (Ordre de division). Pour un alphabet A , l'ordre de division \trianglelefteq est le plus petit pré-ordre de A^* satisfaisant les propriétés suivantes :

- sous-terme : $\forall a \in A, \forall u \in A^*, u \triangleleft au$,
- monotonie : $\forall u, v \in A^*, \forall a \in A, u \triangleleft v \Rightarrow au \triangleleft av$.

Théorème 7.

1. Pour tout alphabet fini A , (A^*, \trianglelefteq) est un bel ordre partiel (Higman [55]).
2. Le type d'ordre maximal de (A^*, \trianglelefteq) est $\omega^{|A|-1}$ (De Jongh et Parikh [71]).

Tout ordre de terminaison totale sur les mots est donc de type d'ordre inférieur à $\omega^{\omega^{\omega}}$ et les dérivations de réécriture peuvent s'interpréter comme des suites ordinales décroissantes de $\omega^{\omega^{\omega}}$. Dans ce cas, on peut en mesurer la longueur grâce aux hiérarchies de fonctions indexées par les ordinaux, et plus précisément grâce à la hiérarchie de Hardy.

Définition 7 (Hiérarchie de Hardy [52]). Pour tout ordinal α et tout entier n , on considère la suite ordinale décroissante $(\alpha_i)_{i \in \mathbb{N}}$ définie par

$$\begin{aligned} \alpha_0 &= \alpha, \\ \alpha_{i+1} &= P_{n+i}(\alpha_i) \text{ (} P_{n+i} \text{ est le prédécesseur donné par une suite fondamentale pour } \alpha_i \text{).} \end{aligned}$$

La suite s'arrête quand elle atteint 0. Nous l'appelons une suite de Hardy. La fonction de Hardy $\mathcal{H}_\alpha : \mathbb{N} \rightarrow \mathbb{N}$ associe à un entier n la longueur de la suite de Hardy initiée par (α, n) .

La classe des fonctions *multiplées récursives* est exactement la famille des fonctions de Hardy indexées par les ordinaux de $\omega^{\omega^{\omega}}$ (Robbin [111]). Cette propriété est à l'origine du théorème suivant, qui fait le lien entre la longueur des mauvaises suites pour l'ordre de division de Higman et son type d'ordre maximal, $\omega^{\omega^{\omega}}$, via la hiérarchie de Hardy.

Théorème 8 (Cichon and Tahhan Bittar [21]). Soit A un alphabet fini et $k \in \mathbb{N}$. Pour tout mot u de A^* , on note $|u|$ la taille de u . Il existe une fonction multiple récursive ϕ telle que toute suite $(u_i)_{i \in \mathbb{N}}$ de A^* satisfaisant

- $\forall i, j \in \mathbb{N}, i < j \Rightarrow \neg(u_i \trianglelefteq u_j)$,
- $\forall i \in \mathbb{N}, |u_i| \leq |u_0| + k \times i$,

soit de longueur bornée par $\phi(|u_0|)$.

Ce résultat fournit incidemment une majoration multiple récursive de la longueur de dérivation d'un système de réécriture de mots qui termine par terminaison totale. Il se trouve que, de façon un peu inattendue, cette borne est essentiellement optimale, c'est-à-dire que les systèmes de réécriture de mots qui terminent totalement épuisent la classe des fonctions récursives multiples en terme de complexité dérivationnelle¹. Deux arguments au moins rendent ce résultat non intuitif. En premier lieu, les hypothèses du théorème sont plus faibles que la terminaison totale engendrée par un système de réécriture : elles ne supposent pas que le système de réécriture soit *fini*, par exemple. De plus, dans le cas des mots, les ordres connus garantissent une longueur de dérivation primitive récursive, bien moins élevée dans la hiérarchie de Grzegorzczuk que les fonctions multiples récursives.

¹Cette question est liée à la résolution du problème 81 de la *RTA list of open problems* (<http://www.lsv.ens-cachan.fr/rtaloop/problems/81.html>).

Théorème 9 (Touzet [133]). *Pour toute fonction multiple récursive f , il existe un système de réécriture de mots fini \mathcal{R} qui termine totalement et tel que f est dominée par $Dl_{\mathcal{R}}$.*

Le principe de la démonstration est de simuler les suites de Hardy pour $\omega^{\omega^{\omega}}$ par un système de réécriture de mots fini qui termine totalement. Nous illustrons son cheminement par un exemple. Soit \mathcal{S} , la suite de Hardy initiée par $(\omega^2, 2)$.

$$(\omega^2, 2) \mapsto (\omega 2, 3) \mapsto (\omega + 3, 4) \mapsto (\omega + 2, 5) \mapsto (\omega + 1, 6) \mapsto (\omega, 7) \mapsto (7, 8)$$

On commence par coder les ordinaux de $\omega^{\omega^{\omega}}$ par des mots. Pour cela, l'ordre LPO sur l'alphabet $\{a_0, \dots, a_i\}$ fournit un système de notation \mathcal{O} pour ω^{ω^i} :

$$\begin{aligned} \mathcal{O} : \quad \omega^{\omega^{\omega}} &\rightarrow A^* \\ 0 &\mapsto \varepsilon \\ \beta + 1 &\mapsto a_0 \mathcal{O}(\beta) \\ \gamma + \omega^{\omega^i} \beta &\mapsto a_{i+1} \mathcal{O}(-1 + \beta) \mathcal{O}(\gamma) \end{aligned}$$

Par exemple, l'entier n est représenté par a_0^n , ω par a_1 , $\omega + n$ par $a_0^n a_1$, $\omega + \omega$ par $a_1 a_0$, ω^2 par $a_1 a_1$, ou $\omega^{\omega} = a_2$. La suite de Hardy \mathcal{S} devient

$$(a_1 a_1, 2) \mapsto (a_1 a_0, 3) \mapsto (a_0^3 a_1, 4) \mapsto (a_0 a_0 a_1, 5) \mapsto (a_0 a_1, 6) \mapsto (a_1, 7) \mapsto (a_0^7, 8)$$

Il reste à transformer chaque couple (u, n) en un mot. On introduit simplement un nouveau symbole $|$ pour représenter les entiers en notation unaire : $|^n$ pour n , que l'on concatène à u . La suite de Hardy précédente s'écrit ainsi

$$||a_1 a_1 \mapsto |||a_1 a_0 \mapsto ||||a_0^3 a_1 \mapsto |||||a_0 a_0 a_1 \mapsto |||||a_0 a_1 \mapsto |||||a_1 \mapsto |||||a_0^7$$

À partir de là, on définit une famille de systèmes de réécriture $(\mathcal{R}_i)_{i \geq 0}$ qui permettent de simuler les suites de Hardy pour $\omega^{\omega^{i-1}}$. Le détail des règles est donné en figure 1.1 et la figure 1.2 montre un exemple complet de dérivation. Le cœur de la preuve est constitué du lemme 1.

Lemme 1 (Touzet [133]). *Pour tout $i \in \mathbb{N}$, le système de réécriture de mots \mathcal{R}_i réduit sous un ordre de terminaison totale de type d'ordre $\omega^{\omega^{i-1}}$, et $Dl_{\mathcal{R}_i}$ domine la fonction de Hardy indexée par $\omega^{\omega^{i-1}}$.*

1.3.3 La réécriture de termes

La construction sur les mots de la section 1.3 a été ensuite reprise et étendue aux systèmes de réécriture de termes par Lepper, en construisant un système de notation pour $\bar{\phi}_{\Omega^{\omega}}(0)$ sur une signature avec des symboles de fonction d'arité quelconque avec la notation de Bachmann [108, 123]. L'ordinal $\bar{\phi}_{\Omega^{\omega}}(0)$, le petit ordinal de Veblen, est le type d'ordre maximal du plongement homéomorphique de Kruskal [119]. Les développements du codage sont ensuite identiques à ceux du théorème 9.

Théorème 10 (Lepper [86]). *Pour tout ordinal $\alpha < \bar{\phi}_{\Omega^{\omega}}(0)$, il existe un système de réécriture de termes fini dont la longueur de dérivation est plus que α -récursive.*

$$\begin{aligned}
\mathcal{R}_0 & \left\{ \begin{array}{ll} a_0 \rightarrow \circ & (0,1) \\ \circ \rightarrow \bullet | & (0,2) \\ \bullet | \rightarrow | \bullet \bullet & (0,3) \\ | \circ \rightarrow \circ | & (0,4) \\ \bullet \rightarrow \varepsilon & (0,5) \end{array} \right. \\
\mathcal{R}_{i+1} = \mathcal{R}_i \cup & \left\{ \begin{array}{ll} \bullet a_{i+1} a_0 \rightarrow k_{i+1} a_{i+1} & (i+1,1) \\ \bullet a_{i+1} \rightarrow a_{i+1} \bullet & (i+1,2) \\ \bullet k_{i+1} \rightarrow k_{i+1} a_i & (i+1,3) \\ a_{i+1} \circ \rightarrow \circ a_{i+1} & (i+1,4) \\ k_{i+1} \rightarrow \circ & (i+1,5) \\ \bullet a_{i+1} \rightarrow k_{i+1} & (i+1,6) \end{array} \right.
\end{aligned}$$

FIG. 1.1: Systèmes de réécriture de mots \mathcal{R}_i ($i \in \mathbb{N}$) pour les suites de Hardy. L'indice ordinal de la fonction de Hardy est codé sur $\{a_0, \dots, a_i\}^*$, l'argument entier par un mot de $\{| \}^*$. Les symboles \circ, \bullet, k_i sont des caractères supplémentaires qui permettent de faire fonctionner la machinerie.

Ce résultats épuise la majoration pour les mauvaises suites du plongement homéomorphe formulée par Weiermann [145].

D'un point de vue pratique, les théorèmes 9 et 10 sur les mots et sur les termes montrent que les ordres de terminaison actuels n'exploitent pas toute l'expressivité des ordres de terminaison totale. Il y a également un intérêt méthodologique. Cela remet en question le *principe de la croissance lente*, que l'on pourrait résumer ainsi :

La complexité dérivationnelle d'un système de réécriture qui termine totalement peut se déduire du type d'ordre de l'ordre de terminaison par application de la hiérarchie à croissance lente.

Cette thèse a été postulée par Cichon [19] et développée notamment par Weiermann [99, 144]. Elle est avérée avec les ordres MPO et LPO :

	MPO	LPO
Type d'ordre	$\bar{\phi}_\omega(0)$	$\bar{\phi}_{\Omega\omega}(0)$
Longueur de dérivation	<i>Réc. primitive</i>	<i>Réc. multiple</i>

Les théorèmes 9 et 10 montrent qu'il n'est pas possible de la généraliser à tous les ordres de terminaison total, et inviteraient à formuler un principe alternatif, basé sur la hiérarchie de Hardy, qui est de croissance plus rapide que la hiérarchie à croissance lente.

La complexité dérivationnelle d'un système de réécriture qui termine totalement peut être majorée par la fonction de Hardy indexée par le type d'ordre de l'ordre de terminaison.

Enfin, l'expressivité de la terminaison totale sur les termes pour la définition de fonctions, avec des systèmes de réécriture de constructeurs, est ouverte. Sur les mots, les systèmes de constructeurs

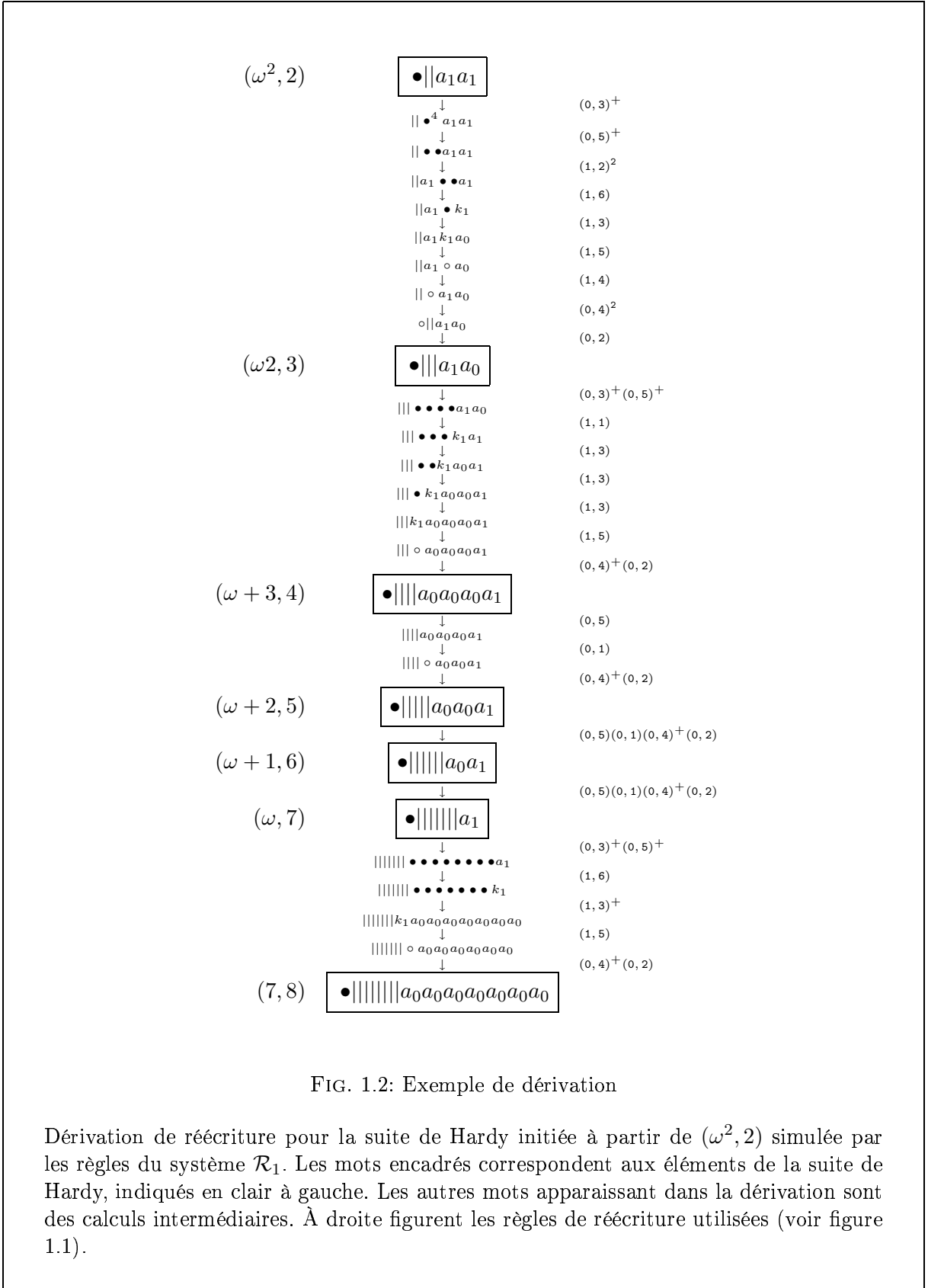


FIG. 1.2: Exemple de dérivation

Dérivation de réécriture pour la suite de Hardy initiée à partir de $(\omega^2, 2)$ simulée par les règles du système \mathcal{R}_1 . Les mots encadrés correspondent aux éléments de la suite de Hardy, indiqués en clair à gauche. Les autres mots apparaissant dans la dérivation sont des calculs intermédiaires. À droite figurent les règles de réécriture utilisées (voir figure 1.1).

permettent de décrire les fonctions primitives récursives seulement, ce qui indique que la classe de complexité est plus petite que pour les systèmes quelconques.

Chapitre 2

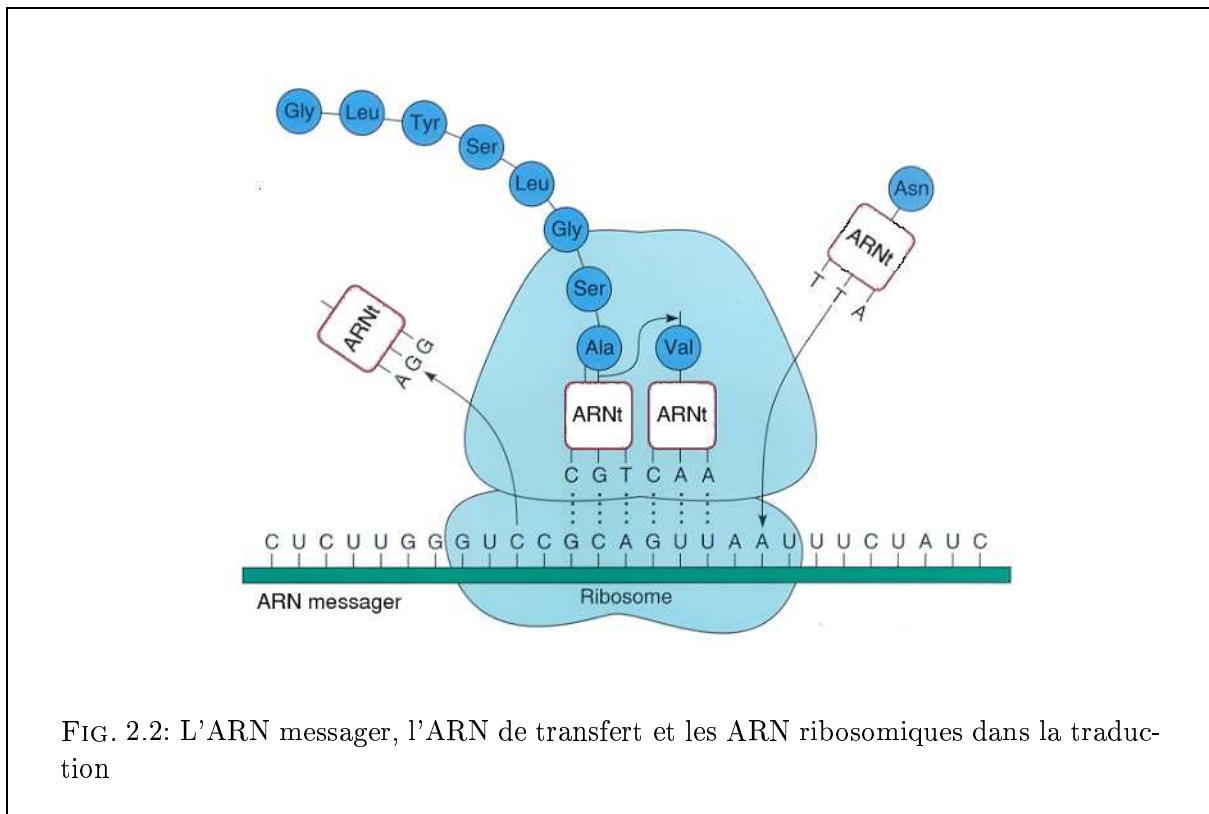
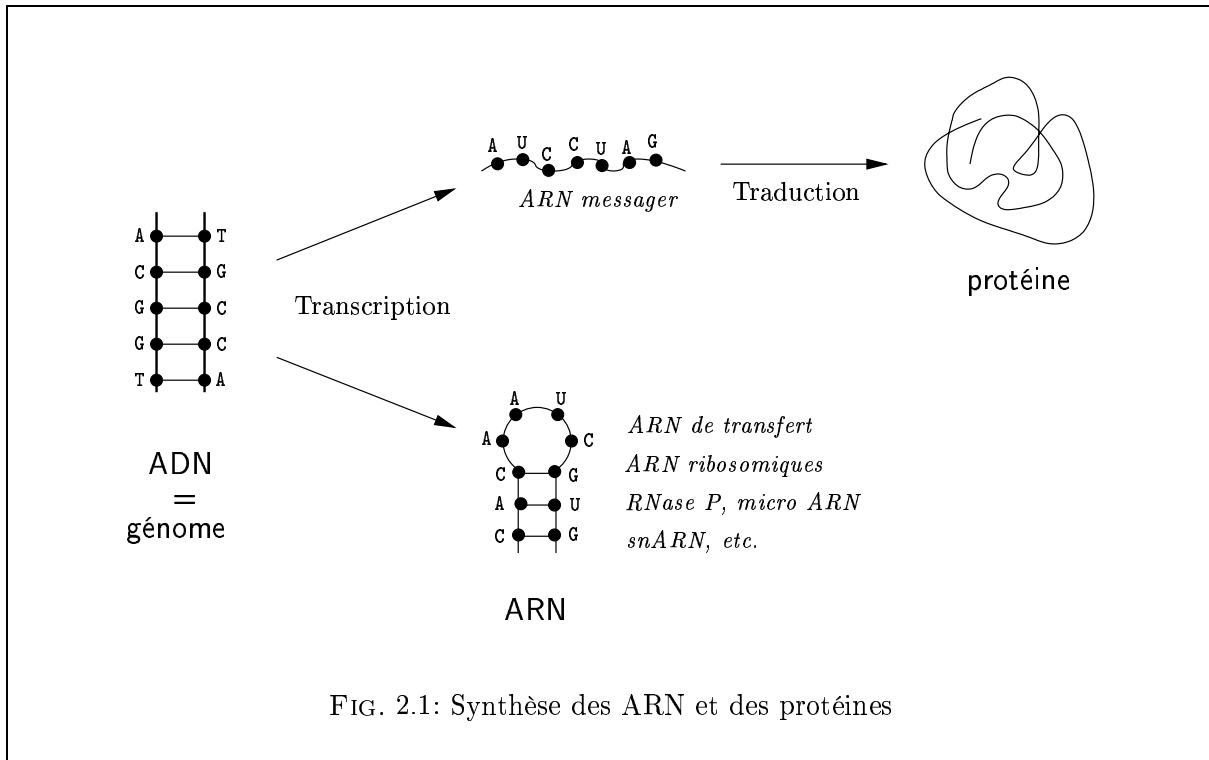
La bio-informatique de l'ARN

Le deuxième thème de ce mémoire est l'algorithmique de l'ARN, que nous abordons à travers deux problèmes dans les chapitres 3 et 4 : la comparaison d'ARN et la prédiction de structures. Les ARN sont des molécules dont le traitement informatique posent des problèmes combinatoires spécifiques en raison de la formation de motifs structurels impliquant des interactions à longue portée au sein de la séquence. À titre d'introduction, ce chapitre présente à grands traits la biologie de l'ARN, les relations entretenues par la séquence et la structure et la modélisation de la structure. Suivant les précautions d'usage, c'est une présentation basique à l'usage du non-spécialiste et déformée par le prisme des problèmes de bio-informatique qui vont suivre.

2.1 L'ARN dans la cellule

Chez les organismes vivants, les mécanismes de la cellule sont orchestrés par trois grands types de molécules : l'ADN (acide désoxyribonucléique), l'ARN (acide ribonucléique) et les protéines. Schématiquement, l'ADN porte l'information génétique, transmise au fil des générations cellulaires, et les protéines expriment cette information, à partir de la transcription et de la traduction des gènes (figure 2.1). Dans ce schéma, le rôle de l'ARN est multiple et diversifié.

L'ARN est d'abord souvent décrit pour son rôle essentiel dans la machinerie de la synthèse protéique. Il intervient tant au niveau de la transcription, avec les ARN messagers, qu'au niveau de la traduction avec les ARN de transfert et les ARN ribosomiques. La figure 2.2 illustre ce processus. À la différence des ARN messagers, les ARN de transfert et les différents ARN ribosomiques sont fonctionnels en tant qu'ARN, sans être traduits en protéine. Ce sont ces ARN, appelés *gènes à ARN* ou *ARN non codants* qui vont motiver les travaux de ce mémoire. On sait maintenant que le champ d'action des ARN non-codants dépasse des exemples historiques de l'ARN de transfert et des ARN ribosomiques. Les deux articles de revue *Non-coding RNA genes and the modern RNA world* [32] et *Breakthrough of the year - small RNAs make big splash* [24] ont à cet égard des titres explicites. Une première découverte décisive, couronnée par le prix Nobel de chimie, date des années 80 avec la mise en évidence que l'ARN possédait des propriétés catalytiques à l'égal des protéines, avec les introns de groupe I. Cette nouvelle fonction a ouvert la voie aux *ribozymes*, avec les RNase P (ribonucléases P), les introns de groupe II, et même les ARN ribosomiques. Plus récemment, l'étude des génomes eucaryotes a révélé l'existence de nombreuses petites molécules d'ARN insoupçonnées. Cela comprend les petits ARN nucléaires, impliqués dans l'épissage, les petits ARN nucléolaires, qui participent aux modifications chimiques du ribosome, les micro-ARN, qui agissent sur l'inhibition de l'expression de gènes.



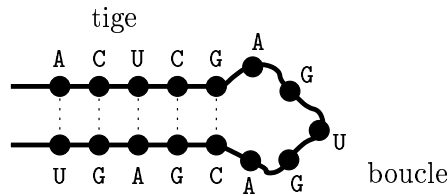
Ces observations réévaluent largement le rôle stratégique des gènes à ARN et permettent de penser que ceux-ci sont des acteurs indispensables de l'expression et de la régulation des gènes aux côtés des protéines.

2.2 Structure de l'ARN

La capacité des ARN à intervenir dans des processus métaboliques variés est liée à ses facultés de repliement et à la réactivité de ses bases au sein des structures ainsi formées. En comparaison avec l'ADN dont il est pourtant chimiquement proche, l'ARN jouit d'une grande souplesse structurale.

La chaîne principale de l'ARN est composée de sucres, les riboses, liés entre eux par des phosphates. À chaque ribose est attachée une base, purique ou pyrimidique, qui sont les constituants fondamentaux. L'ARN a quatre bases différentes : l'adénine, la guanine, la cytosine et l'uracile. Les trois premières participent également à la constitution de l'ADN, et l'uracile remplace la thymine comme base complémentaire de l'adénine.

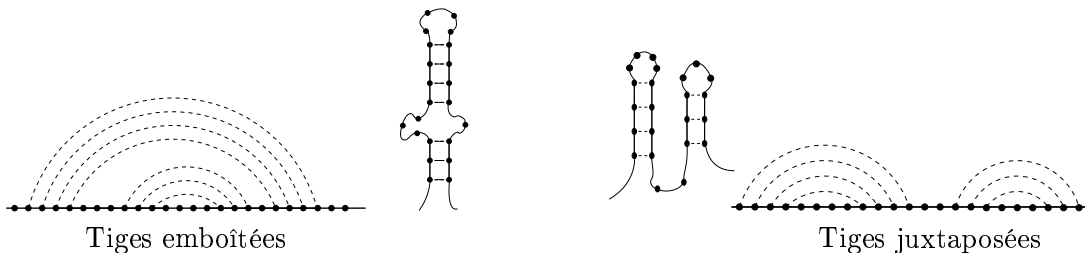
À la différence notable de l'ADN, l'ARN est une molécule *simple brin*. Cela autorise les bases de la molécule à s'apparier entre elles, formant des liaisons hydrogène.



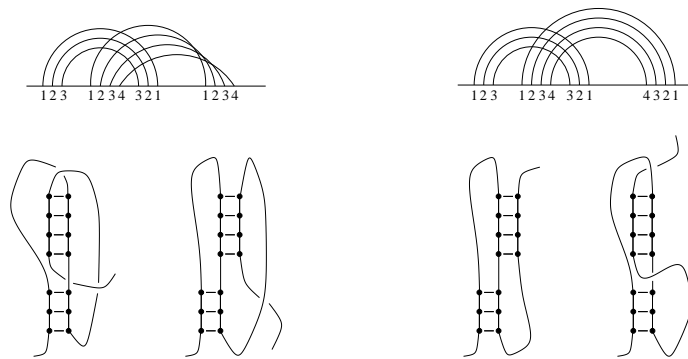
Les appariements les plus courants, canoniques, sont les appariements de Watson-Crick (A-U, C-G), à l'image de l'ADN, et les appariements faibles (U-G). Mais pratiquement tous les appariements sont observés [101]. Les appariements ne sont pas isolés. Ils se forment de manière contigue, pour construire des *tiges*, stabilisées par empilement avec les appariements adjacents. Les régions restées non appariées forment des *boucles*. Cela conduit à une configuration spatiale qui conditionne la fonction de la molécule. Cette structure est décrite par une classification en quatre niveaux hiérarchiques. La figure 2.3 montre les quatre niveaux de structure pour l'ARNt.

La *structure primaire* est la suite de nucléotides, orientée de 5' en 3', qui s'obtient par séquençage. La structure primaire est généralement symbolisée par un mot sur l'alphabet {A,U,C,G}.

La *structure secondaire* est l'ensemble des appariements entre bases, qui sont emboîtés \ll ou juxtaposés \triangleleft : $(i, j) \ll (k, l)$ si $\{k \leq i \text{ et } j \leq l\}$ et $(i, j) \triangleleft (k, l)$ si $j \leq k$.



La *structure tertiaire* inclut les appariements résiduels : pseudo-nœuds (appariements chevauchants), triplets (appariements à trois), quartets (à quatre), appariements isolés.



Exemples de pseudo-nœuds

Enfin, la *structure spatiale* désigne la configuration de la molécule dans l'espace. A l'instar de l'ADN, les tiges s'enroulent en doubles hélices. La double hélice de l'ARN diffère toutefois de celle de l'ADN en raison de la présence du ribose, au lieu du désoxyribose, dans le squelette sucre-phosphate de la chaîne. C'est une double hélice de type A, et non de type B, orientée également vers la droite, mais plus courte et plus large, avec 11 paires de bases par tour, et non 10. Comme pour les protéines, le repliement est dirigé par des forces hydrophobes [122]. Il est également gouverné par les molécules – ions et surtout protéines – avec lesquelles l'ARN va interagir [132]. Dans la grande majorité des cas, les ARN ne restent pas nus. Ils s'assemblent avec des protéines pour former des complexes fonctionnels. C'est le cas des ribosomes ou des ribonucléases.

2.3 Le statut privilégié de la structure secondaire

2.3.1 L'analyse informatique des ARN

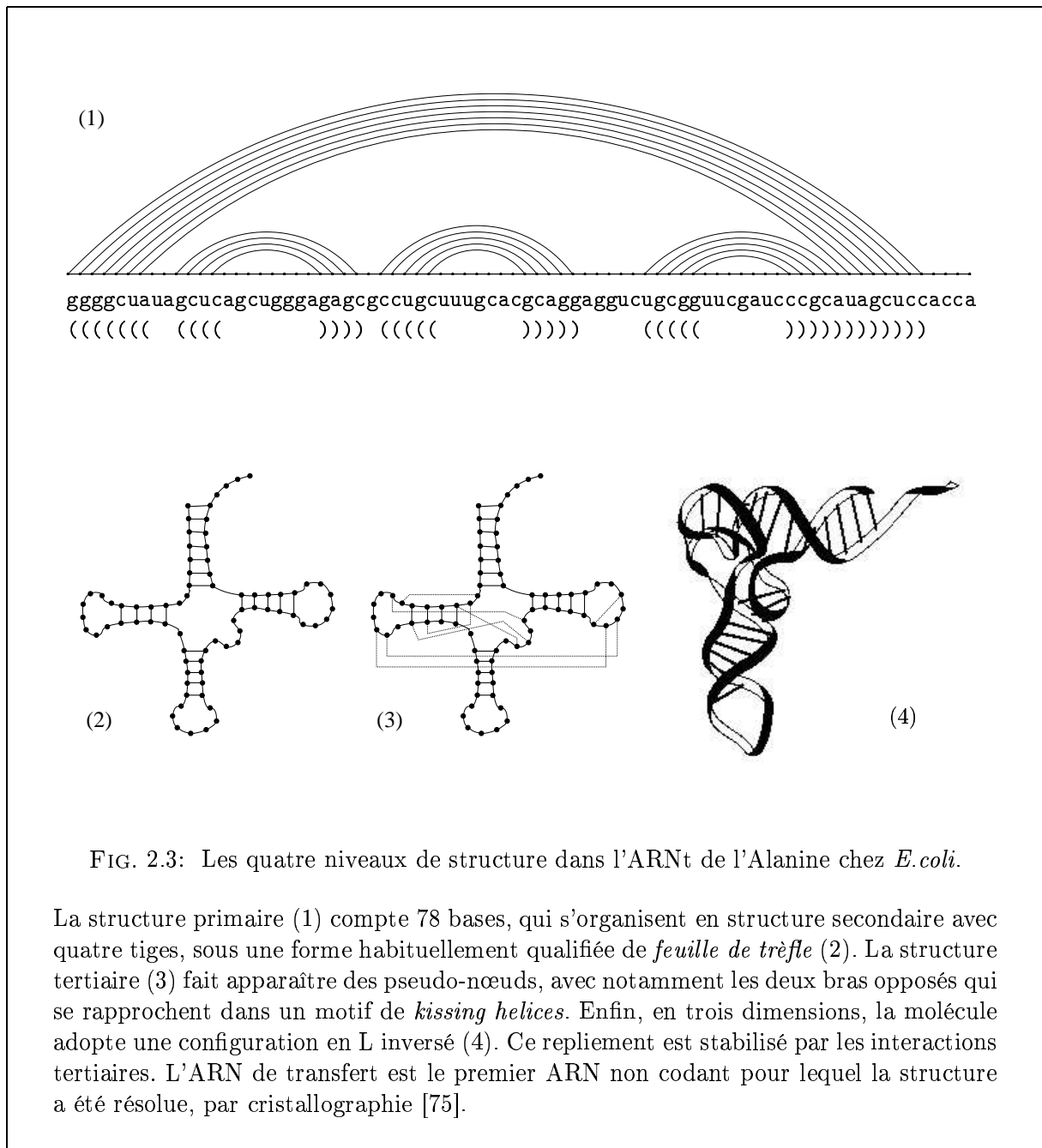
L'ARN est donc un objet combinatoire complexe, qui pose des problèmes spécifiques de compréhension des relations entre la séquence et la structure. Il est généralement admis que la fonction d'une molécule est davantage portée par sa structure que sa séquence. Dans cette perspective, l'analyse des ARN est à l'origine des thèmes de recherche suivants.

Prédiction de structure. On sait établir la structure tridimensionnelle des ARN par analyse de spectres de diffraction aux rayons X obtenus sur des cristaux moléculaires. Mais cela reste une activité exceptionnelle, beaucoup plus lourde que le seul séquençage de la molécule. Dès les prémices de la bio-informatique ont été proposés des algorithmes pour inférer la structure des ARN à partir de la connaissance de la structure primaire [103, 118].

Comparaison. Il s'agit de comparer des molécules d'ARN en tenant compte des différents niveaux de structure. La comparaison peut s'appliquer à des structures exactes ou des structures prédites. Elle permet de mettre en lumière les motifs conservés, motif s'entendant ici au sens nucléique et au sens structural, ou les régions de divergence.

Localisation. Quand on a modélisé la structure d'une famille d'ARN, on peut en dériver des stratégies pour localiser les occurrences des gènes de cette famille sur un génome. Des programmes dédiés ont ainsi été développés pour la recherche d'ARNt [92, 34] ou de RNase P [22]. Quelques algorithmes existent pour des descripteurs universels, tels que Palingol [9] ou Erpin [44].

Prédiction de gènes. Il s'agit de la détection des gènes à ARN sur un génome, sans connaissance a priori de la structure des gènes impliqués. C'est la contrepartie ribonucléique des méthodes de



```

((((((( ((( ( ))) ( ( ((( ))) ) ) ((( ( )))))))))))
GTCCGAATAGCTCAGCTGGATAGAGCAA T AGCCTTCTAAGCT A TCGGTCGGGGTTCGAATCCCTCTTCGGACGCCA
GCACTCGTAGCTTAAC-GGATAAAGCAT C TGACTACGGATCA G AAGGTTGCAGGTTTGAATCCTGCCGAGTGCA--
GTCCACGTAGCTCAGCAGGATAGAGCAC A GGATTCCTAATCC T GGGGTTGGAGGTTTGAATCCTCTCGTGGACACCA
GCGCCCGTAGCTCAATTGGATAGAGCGT T TGACTACGGATCA A GAGGTTATGGGTTTGAATCCTCTCGGGCGCG--
GCACCCATAGCGCAACTGGATAGAGTGT C TGACTACGAATCA G AAGGTTGTAGGTTTGAATCCTACTGGGTGCA--
GCGCCCGTAGCTCAATTGGATAGAGCGT T TGACTACGGATCA A AAGGTTAGGGTTCGACTCCTCTCGGGCGCGCCA
GCGCCCTTAGCTCAGTTGGATAGAGCAA C GACCTTCTAAGTC G TGGCCCGCAGGTTTGAATCCTGCAGGGCGCGCCA

```

FIG. 2.4: Mutations compensatoires

La figure représente un alignement de sept séquences d'ARN de transfert, pour lesquelles la structure commune est indiquée, en première ligne. Les deux colonnes isolées participent à un même appariement et font apparaître des mutations compensatoires.

prédiction de gènes pour les protéines basées sur le modèle statistique de la table d'usage des codons, par exemple. Pour l'ARN, ce problème est très partiellement résolu, faute de modèle.

Toutes ces questions sont généralement traitées au niveau de la structure secondaire, plutôt qu'à celui de la structure tertiaire, voire tri-dimensionnelle. Cela s'explique par le rôle prépondérant de la structure secondaire et par sa relative maniabilité.

2.3.2 Comment évoluent les ARN

Dans les gènes codant pour les protéines, la pression de sélection s'exerce essentiellement sur la séquence traduite. On observe ainsi des mutations silencieuses au niveau du gène, qui n'auront pas d'effet sur l'acide aminé après traduction. Par exemple, le troisième nucléotide, qui est le moins contraint dans le code génétique, a un taux de variabilité plus important que les deux positions précédentes.

On peut faire un parallèle avec l'évolution des gènes à ARN. L'équivalent des mutations silencieuses sont en quelque sorte les *mutations compensatoires*. Le principe est le suivant. Quand une base impliquée dans un appariement mute, la base complémentaire mute également, pour maintenir l'appariement et préserver la structure. La figure 2.4 montre un alignement de sept séquences d'ARNt, où apparaissent des mutations compensatoires.

Cela signifie que ce sont essentiellement les niveaux de structure secondaire et tertiaire qui supportent la pression de sélection, et non la structure primaire. Par exemple, la structure de l'ARN de transfert en feuille de trèfle, présentée en figure 2.3 est conservée dans les différents règnes. C'est d'ailleurs cette propriété qui a permis d'inférer correctement la structure à partir d'une famille d'ARNt [89], plusieurs années avant la détermination expérimentale. Quand on regarde les séquences, le pourcentage d'identité moyen est de 42% seulement (RFAM [49], entrée RF0005).

La figure 2.5 donne un second exemple de conservation. Il s'agit de RNase P de deux protéobactéries de la sous-division delta/epsilon. Les deux structures secondaires sont quasiment identiques, alors qu'il y a une forte divergence au niveau de la structure primaire. Quand on considère des organismes plus éloignés, une étude phylogénétique entre les eucaryotes et les bactéries montre que la structure secondaire formant le cœur catalytique de la molécule reste

conservée [40].

2.3.3 Structure secondaire vs structure tertiaire

La structure primaire est donc peu informative pour les molécules d'ARN, et une analyse pertinente doit prendre en compte les niveaux de structures supérieurs. Dans ce mémoire, nous accordons un rôle prépondérant à la structure secondaire.

Le premier argument est numérique : la structure secondaire couvre la majorité des appariements, souvent plus de 90% dans les gènes de plus de 60 bases. C'est donc une approximation satisfaisante de la structure tertiaire, tout en offrant l'avantage d'une structure combinatoire plus simple. D'un point de vue algorithmique, la prise en compte des interactions tertiaires et *a fortiori* du repliement en trois dimensions change la nature des questions à traiter. Que ce soit pour la prédiction, la comparaison, la majorité des problèmes deviennent NP-complets [93, 69, 139].

Cette distinction de la structure secondaire trouve également une justification biologique. En effet, les niveaux de structure secondaire et tertiaire peuvent être séparés expérimentalement [132, 147, 153], et donc observés indépendamment. De plus, la cinétique du repliement fait que la structure secondaire est un préalable à la structure tertiaire. Une fois la structure secondaire formée, les tiges s'agencent dans l'espace pour former la structure tridimensionnelle. C'est à cette étape que se forment les pseudo-nœuds et les autres interactions tertiaires. Il est possible que le repliement final occasionne des remaniements au sein de la structure secondaire initiale et que la structure secondaire finale (déduite théoriquement à partir de la structure tertiaire) ne corresponde donc pas exactement à la structure secondaire initiale [153]. Mais beaucoup de tiges sont communes. La structure secondaire constitue l'*échaffaudage* à partir duquel la structure tertiaire va se bâtir [104].

Ces propriétés expliquent que bien que la structure secondaire ne recèle pas toute l'information, elle soit conservée au fil de l'évolution. On peut donc voir la structure secondaire comme une trace de la structure tridimensionnelle, trace accessible au calcul. Cela rend son étude légitime et c'est dans cette perspective que nous travaillons. La comparaison de structures secondaires permet ainsi de fournir de meilleurs alignements que sur la seule structure primaire, idéalement des alignements compatibles avec la structure tridimensionnelle. Quant à la prédiction, l'inférence de la structure secondaire commune pour une famille d'ARN homologues permet de capturer l'essentiel de la structure tertiaire.

2.3.4 Représentation de la structure secondaire

Pour pouvoir manipuler les structures secondaires d'ARN, il faut se donner une représentation symbolique. Cela doit être une description informative, qui exprime la contribution de la structure à la fonction, et qui capture l'homologie entre plusieurs ARN. Ce problème est non-trivial, et reste encore ouvert à toutes les réflexions. Les solutions les plus courantes font appel à des arbres ordonnés.

En effet, la définition de la structure secondaire, qui exclut les croisements entre les bases appariées, invite naturellement à penser à une structure arborescente. C'est une expression bien parenthésée, où une parenthèse ouvrante correspond au nucléotide en 5' et la parenthèse fermante au nucléotide en 3' d'un appariement. Cela revient à considérer des forêts ordonnées. Une feuille de l'arbre correspond à une base libre, étiquetée par un élément de $\{A, U, C, G\}$, et un nœud interne à un couple de bases appariées, étiqueté par un élément de $\{A, U, C, G\} \times \{A, U, C, G\}$. Les labels des nœuds intègrent ainsi les informations de structure primaire. La figure 2.6-1 montre

l'arbre associé à l'ARN de transfert de la figure 2.3. En terme d'édition, on peut définir trois opérations élémentaires, à l'image de celles définies pour l'alignement de séquences d'ADN ou de protéines.

- *Substitution* : on remplace un label par un autre.
- *Insertion* : on ajoute un nœud à un arbre.
- *Délétion* : on supprime un nœud à un arbre. Les fils de ce nœud deviennent ainsi les fils du père.

Pour chacune de ces opérations, les modifications sur les nœuds internes correspondent à des altérations de bases appariées. La substitution s'interprète alors comme la mutation d'une paire de bases, avec mutation compensatoire, dans les tiges. Les modifications sur les feuilles correspondent à des boucles.

Ce premier modèle interdit toute opération d'édition entre les bases libres et les bases appariées. D'un point de vue biochimique, on ne peut pas exprimer simplement la cassure d'un pont hydrogène entre deux bases. On peut intégrer cette contrainte en enrichissant l'ensemble des opérations d'édition [68]. Une autre variation, proposée dans [56], est de représenter chaque paires de bases par un triplets de nœuds : un nœud interne pour l'appariement, accompagné de deux feuilles pour les deux bases (figure 2.6-2). Chaque base est ainsi exactement représentée par une feuille. Ce modèle autorise également les recompositions de paires.

Un inconvénient commun à ces deux codages est qu'ils raisonnent au niveau d'une base, ou d'une paire de bases. Ils ne reflètent pas de manière intrinsèque les propriétés de voisinage entre bases libres et au sein des tiges. Ainsi, il n'y a pas de différence, en terme d'édition, entre supprimer une tige formée par l'empilement de trois paires, ou supprimer trois paires au sein d'une tige plus longue. Le niveau de description est trop détaillé pour être informatif. Ces codages autorisent également des événements évolutifs improbables, tels que la combinaison de tiges.

Un meilleur modèle doit pouvoir rendre compte de l'organisation en tiges et en boucles de la structure secondaire. Une première solution est de compacter l'arbre initial verticalement et horizontalement [37]. Un nœud interne correspond alors à une tige, et une feuille à une boucle (voir Figure 2.6-3). Cette représentation ménage la possibilité d'intégrer dans le système de score, sans coût algorithmique, la différence entre retirer un appariement dans une tige (substitution dans un nœud interne), et supprimer une tige, ce qui modifie la structure de la molécule.

La structure d'une tige et son encombrement dans l'espace dépendent essentiellement de sa longueur. En revanche, les contributions des bases libres à la structure finale sont plus sensibles. Les boucles participent à la direction spatiale des tiges en fonction de leur localisation dans la molécule et de leur degré. On classe ainsi les boucles en quatre types [124] : les boucles terminales H, qui forment des structures en épingle à cheveux avec la tige associée, les boucles internes I, qui s'intercalent dans une tige, les renflements B et les boucles à embranchement multiple M. Les tiges non terminales sont S (voir figure 2.7). Formellement, la syntaxe d'une structure secondaire « SIBMH » est décrite par la grammaire

$$\left\{ \begin{array}{l} X \rightarrow sY\bar{s}|h \\ Y \rightarrow mT\bar{m}|iX\bar{i}|bX\bar{b} \\ T \rightarrow XT|X \end{array} \right.$$

La figure 2.8 présente un exemple détaillé pour l'ARNr 5S. Cette philosophie de représentation modulaire en éléments structuraux peut être raffinée avec des modèles multi-couches ou multi-échelles [105].

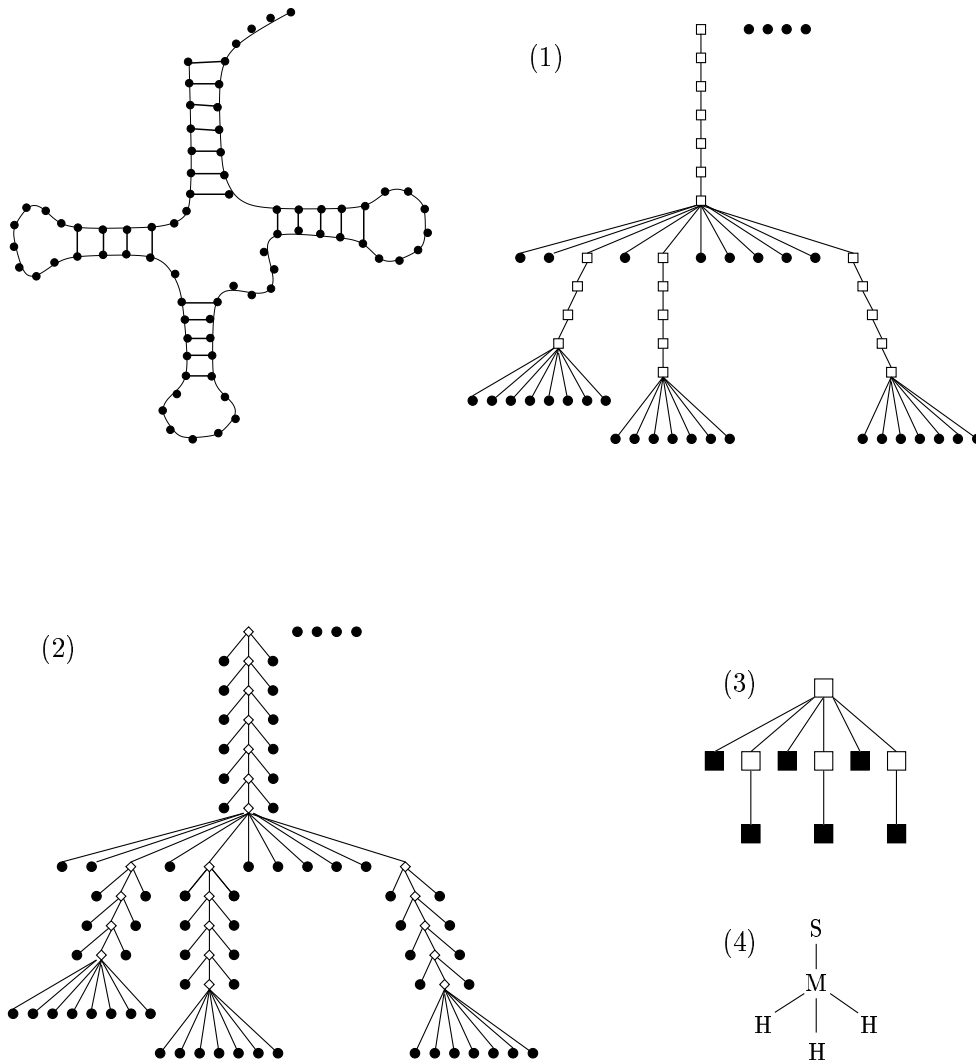
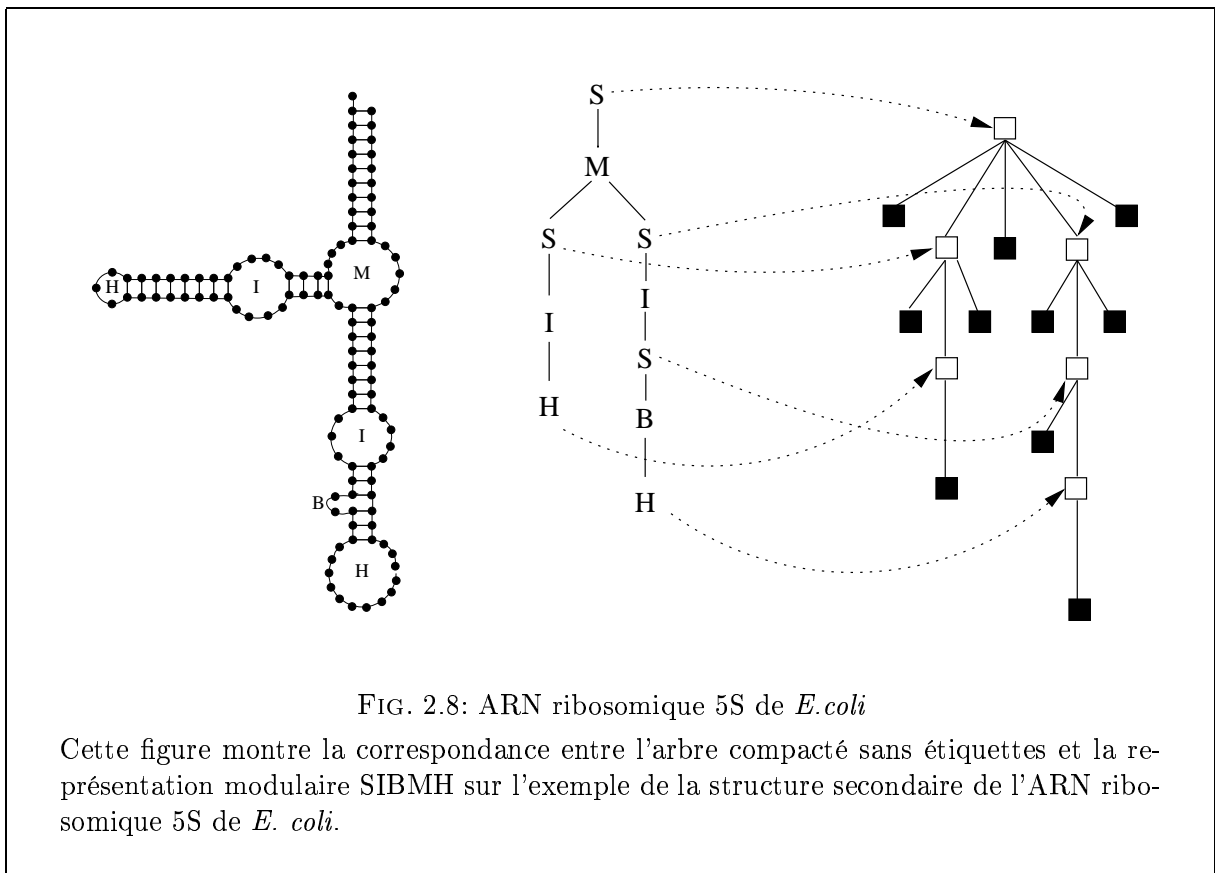
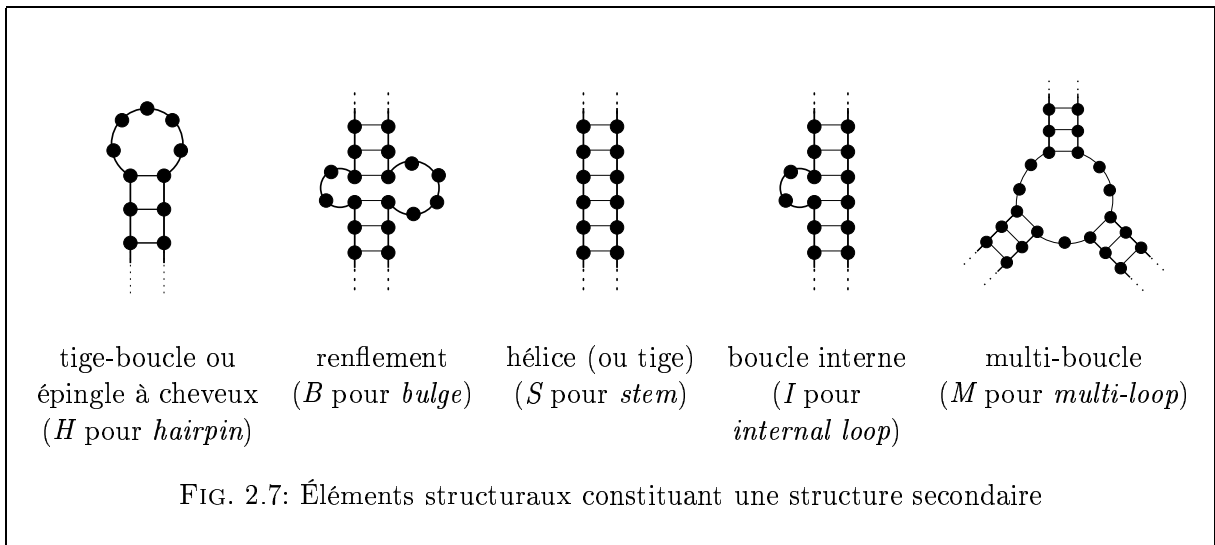


FIG. 2.6: Quatre représentations arborescentes pour la structure secondaire de l'ARN de transfert de l'Alanine chez *E. coli*

Cette figure reprend la structure secondaire de la figure 2.3 codée sous forme d'arbre complet (1), d'arbre avec triplets pour chaque paire de bases (2), d'arbre compacté (3) et avec un codage par modules SIBMH (4).



Chapitre 3

Algorithmes pour la comparaison de molécules d'ARN

La motivation pour comparer syntaxiquement des molécules est que la similitude révèle une proximité de fonction et que les régions conservées sont fonctionnellement importantes. Shapiro est le premier à avoir appliqué ce postulat fondateur de manière systématique à des gènes à ARN [124]. La comparaison peut être également utilisée à des fins de phylogénie [23]. Elle trouve d'autres applications avec des structures prédites, pour décider par exemple si deux séquences pourraient être homologues.

Comment comparer deux ARN ? Pour que la comparaison fasse du sens, il faut l'adresser à un niveau fonctionnel, c'est-à-dire contenant au moins la structure secondaire. La section 2.3.4 montre que les représentations usuelles de la structure secondaire de l'ARN se réduisent à des structures d'arbres. On est donc ramené à un cadre algorithmique de comparaison d'arbres qui dépasse le contexte de la biologie moléculaire. Il y a deux paradigmes pour cela : la distance d'édition et l'alignement, dont nous discutons et dont nous voyons les mérites comparés.

Ce chapitre dresse un panorama assez touffu des principaux algorithmes de comparaison en proposant plusieurs déclinaisons adaptées au contexte de l'ARN : les pénalités de gaps, la comparaison locale, l'intégration de contraintes. L'exposé s'appuie l'introduction du graphe d'édition.

3.1 Arbres et forêts

3.1.1 Notations

Un arbre est un nœud (la racine) connecté à une forêt, c'est-à-dire une suite finie ordonnée d'arbres disjoints. On note $\ell(A_1 \circ \dots \circ A_n)$ l'arbre composé du nœud ℓ connecté à la suite d'arbres A_1, \dots, A_n . Par convention, ε désigne l'arbre vide aussi bien que la forêt vide. Un arbre est une forêt particulière. Nous utilisons les notations suivantes.

- $|F|$ est la taille, c'est-à-dire le nombre de nœuds, de F
- $depth(i)$ est la profondeur du nœud i , l'arbre de référence étant précisé par le contexte,
- $height(F)$ est la hauteur de la forêt F ,
- $deg(i)$ est le degré du nœud i , c'est-à-dire son nombre de fils. Par extension, $deg(F)$ désigne le degré maximal d'un nœud de F ,
- $\#leaves(F)$ est le nombre de feuilles de F .

3.1.2 Distance d'édition

Le modèle de la distance d'édition pour comparer deux arbres, ou deux forêts, a été introduit par Tai [130]. Nous avons défini au chapitre précédent les trois opérations d'édition élémentaires :

- *Substitution* : on remplace un label par un autre : $\ell(A_1 \circ \dots \circ A_n) \rightarrow \ell'(A_1 \circ \dots \circ A_n)$
- *Insertion* : on ajoute un nœud à un arbre : $f \rightarrow \ell(f)$
- *Délétion* : on supprime un nœud à un arbre. Les fils de ce nœud deviennent ainsi les fils du père : $\ell(f) \rightarrow f$

Un *script d'édition* entre deux forêts F et G est une suite d'opérations d'édition transformant F en G . L'ordre dans lequel sont appliquées les opérations d'édition est souvent significatif. Si on assigne un coût à chacune des opérations d'édition, la *distance* entre deux forêts F et G est le coût minimal nécessaire d'un script d'édition qui transforme F en G . À chaque script d'édition correspond un *mapping*, qui est simplement le sous-ensemble de $F \times G$ constitué des couples (i, j) tels que i est substitué en j .

On peut également voir le problème sous l'angle de la *similitude*. Les insertions et les délétions ont des scores négatifs, la substitution un score positif ou négatif suivant les symboles en jeu. La similitude correspond alors au score maximal d'un script d'édition. Par la suite, par confort, nous utiliserons souvent le terme *distance*, même s'il s'agit de similitude, pour faire référence au modèle de comparaison défini par ces trois opérations d'édition. On note $d(F, G)$ la distance, ou la similitude, entre F et G .

3.1.3 Sous-forêts

À l'instar de l'alignement sur les séquences, les algorithmes de comparaison d'arbres travaillent par programmation dynamique, par décompositions successives des forêts considérées. À cet effet, nous définissons plusieurs types de sous-forêt, qui apparaîtront au fil des sections suivantes.

Définition 8 (Sous-arbre, sous-arbre complet, forêt droite, gauche, suffixe, close, spéciale). Soit F , une forêt.

1. Un sous-arbre de F est un sous-graphe connexe de F .
2. Un sous-arbre complet de F est un sous-arbre A tel que pour tout nœud x de A , les descendants de x appartiennent à A . Un sous-arbre complet est ainsi caractérisé par un nœud i , la racine de A . On note $F(i)$ le sous-arbre complet associé à i .
3. L'ensemble des sous-forêts gauches de F est le plus petit ensemble satisfaisant
 - F est une forêt gauche et pour tout nœud i de F , $F(i)$ est une forêt gauche,
 - si $t \circ \ell(g)$ est une forêt gauche, alors $t \circ g$ est également une forêt gauche.
4. De même, l'ensemble des sous-forêts droites de F est le plus petit ensemble satisfaisant
 - F est une forêt droite et pour tout nœud i de F , $F(i)$ est une forêt droite,
 - si $\ell(g) \circ t$ est une forêt droite, alors $g \circ t$ est également une forêt droite.
5. L'ensemble des sous-forêts suffixes de F est le plus petit ensemble satisfaisant
 - F est une forêt suffixe,
 - pour tout nœud i de F , si $F(i) = \ell(f)$, alors f est une forêt suffixe,
 - si $\ell(g) \circ t$ est une forêt suffixe, alors t est également une forêt suffixe.
6. L'ensemble des sous-forêts closes de F est le plus petit ensemble satisfaisant
 - F est une forêt close,

- pour tout nœud i de F , si $F(i) = \ell(f)$, alors f est une forêt close,
 - si $\ell(g) \circ t$ est une forêt close, alors t est également une forêt close,
 - si $t \circ \ell(g)$ est une forêt close, alors t est également une forêt close.
7. L'ensemble $\text{Special}(F)$ des sous-forêts spéciales de F est le plus petit ensemble satisfaisant
- F est une forêt spéciale,
 - pour tout nœud i de F , $F(i)$ est une forêt spéciale,
 - si $\ell(g) \circ t$ est une forêt spéciale, alors $g \circ t$ est également une forêt spéciale,
 - si $t \circ \ell(g)$ est une forêt spéciale, alors $t \circ g$ est également une forêt spéciale.

Les forêts droites, gauches, closes et suffixes et les sous-arbres complets sont des exemples de forêts spéciales. On note $\#\text{left}(F)$ le nombre de sous-forêts gauches, $\#\text{right}(F)$ le nombre de sous-forêts droites, $\#\text{suffixe}(F)$ le nombre de sous-forêts suffixes, $\#\text{close}(F)$ le nombre de sous-forêts closes, et $\#\text{special}(F)$ le nombre de forêts spéciales de F .

Proposition 1. Soit A un arbre.

$$\begin{aligned} \#\text{right}(A) &= \sum(|A(i)|, i \in A) - \sum(|A(j)|, j \text{ est un fils droit}) \\ \#\text{left}(A) &= \sum(|A(i)|, i \in A) - \sum(|A(j)|, j \text{ est un fils gauche}) \\ \#\text{suffixe}(A) &= |A| \\ \#\text{close}(A) &= \sum(\text{deg}(i)(\text{deg}(i) + 1)/2, i \in A) \\ \#\text{special}(A) &= \frac{|A|(|A|+3)}{2} - \sum_{i \in A} |A(i)| \quad (\text{Dulucq et Touzet [30]}) \end{aligned}$$

3.2 Algorithmes pour la distance d'édition entre arbres

3.2.1 Le graphe d'édition

On peut présenter l'algorithme classique de distance entre deux mots comme un problème de chemin de coût optimal dans un graphe. Soient u et v , deux mots à aligner, de longueur m et n , indexés à partir de 0. On associe à u et v un graphe orienté constitué de nœuds de $0..n \times 0..m$. C'est en fait une grille de taille $(n + 1) \cdot (m + 1)$. De chaque nœud (i, j) sont issus trois arcs

$$\begin{aligned} (i, j) &\rightsquigarrow (i + 1, j) && \text{étiqueté par le coût d'une délétion de } u(i) \\ (i, j) &\rightsquigarrow (i, j + 1) && \text{étiqueté par le coût d'une insertion de } v(j) \\ (i, j) &\rightsquigarrow (i + 1, j + 1) && \text{étiqueté par le coût d'une substitution de } u(i) \text{ en } v(j) \end{aligned}$$

Les chemins menant du nœud $(0, 0)$ au nœud (m, n) sont exactement tous les alignements possibles. L'alignement optimal est obtenu en construisant le chemin de score maximal. Ce point de vue est en fait une simple reformulation visuelle de l'algorithme de programmation dynamique usuel [102, 141].

On peut généraliser le graphe d'édition sur les mots à la distance entre arbres. Un mapping entre deux arbres induit un alignement entre les deux mots formés par les notations préfixes de ces deux arbres. Bien sûr, tous les alignements ne sont pas corrects. L'alignement doit être cohérent avec la structure de l'arbre. Si i est substitué à j , alors $A(i)$ et $B(j)$ doivent correspondre dans le mapping. Sur le graphe d'édition, cela signifie que si un chemin passe par un arc diagonal $(i, j) \rightsquigarrow (i + 1, j + 1)$, alors il doit passer par le nœud $(i + l, j + k)$ où l est la taille de $A(i)$ et k est la taille de $B(j)$.

De cette remarque vient l'idée du graphe d'édition dédié à la distance d'édition entre arbres. Les nœuds sont les mêmes, ainsi que les arcs horizontaux et verticaux associés aux opérations de délétion et d'insertion. Seuls les arcs correspondant à une substitution sont modifiés.

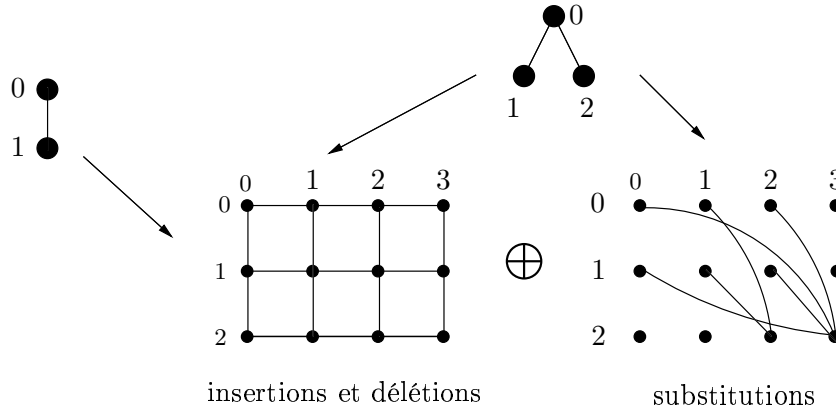


FIG. 3.1: Graphe d'édition pour la distance d'édition entre arbres

$$\begin{aligned}
 (i, j) &\rightsquigarrow (i + 1, j) && \text{étiqueté par le coût d'une délétion de } u(i) \\
 (i, j) &\rightsquigarrow (i, j + 1) && \text{étiqueté par le coût d'une insertion de } v(j) \\
 (i, j) &\rightsquigarrow (i + |A(i)|, j + |B(j)|) && \text{étiqueté par } d(A(i), B(j))
 \end{aligned}$$

La figure 3.1 donne un exemple. Comment retracer le mapping à partir de ce graphe d'édition ? Une première recherche de chemin optimal de $(0, 0)$ vers (m, n) donne les couples de nœuds sans ancêtres appartenant au mapping optimal. Il faut ensuite recommencer récursivement sur les couples de sous-arbres correspondant aux nœuds apparaissant dans cette ébauche de mapping.

Proposition 2. *La donnée du graphe d'édition entre arbres permet de construire le mapping optimal en temps $O(n^3)$ et en espace $O(n^2)$.*

Les algorithmes de distance d'édition présentés dans la suite de cette section permettent tous de calculer le graphe d'édition.

3.2.2 Algorithme de Zhang-Shasha

C'est le premier algorithme efficace qui a été proposé pour la distance entre arbres, qui est mis en œuvre dans le *Vienna RNA Package* [57]. Le principe est de décomposer les deux arbres suivant les opérations d'édition à appliquer. C'est le même mode opératoire que l'algorithme de distance d'édition entre séquences. La première opération concerne nécessairement la racine d'au moins un des deux arbres.

$$d(\ell(f), \ell'(g)) = \max \begin{cases} c_d(\ell) + d(f, \ell'(g)) & \text{délétion de } \ell \\ c_i(\ell') + d(\ell(f), g) & \text{insertion de } \ell' \\ c_s(\ell, \ell') + d(f, g) & \text{substitution de } \ell \text{ en } \ell' \end{cases} \quad (3.1)$$

Cela pose maintenant le problème de la distance pour des forêts. L'insertion et la délétion sont similaires aux formules de récurrence dans le cas des mots. La substitution génère deux nouveaux appels récursifs.

$$\begin{aligned}
 d(t \circ \ell(f), v \circ \ell'(g)) = \\
 \max \begin{cases} c_d(\ell) + d(t \circ f, v \circ \ell'(g)) & \text{délétion de } \ell \\ c_i(\ell') + d(t \circ \ell(f), v \circ g) & \text{insertion de } \ell' \\ d(\ell(f), \ell'(g)) + d(t, v) & \text{substitution de } \ell \text{ en } \ell' \end{cases} \quad (3.2)
 \end{aligned}$$

Les sous-forêts qui apparaissent ainsi au fil du calcul sont soit des sous-arbres, soit des préfixes de sous-arbres : ce sont des forêts gauches. D'après la proposition 1, cela garantit une majoration en $O(n^4)$ en temps. Cette borne est stricte. Considérons les arbres peignes b_n de taille $2n + 1$: chaque nœud interne a exactement deux fils, et le fils gauche est une feuille. Dans ce cas, le nombre de sous-forêts gauches est $(2n^2 + 1)$ pour un arbre. Le comportement en moyenne est toutefois meilleur. On peut montrer que $\#left(A) \leq |A| \min(\#leaves(A), height(A))$ [156], et la complexité en moyenne est $O(n^3)$ [29].

La mise en équations du problème invite à penser à une mise en œuvre par programmation dynamique. On introduit deux types de tables :

- une table principale **Arbre** pour les distances entre sous-arbres (équation 3.1). **Arbre** est une matrice de taille $(m + 1)(n + 1)$ indexée par les nœuds classés par ordre préfixe, qui correspondent aux étiquettes du graphe d'édition.
- une table temporaire **Forêt** pour les distances entre forêts (équation 3.2). On ne construit de table temporaire que pour les couples (i, j) de $A \times B$ tels que i et j ne sont pas tous les deux des fils droits.

On peut ainsi construire avec la table **Arbre** le graphe d'édition en espace $O(mn)$.

Une version alternative de l'algorithme de Zhang-Shasha

Il est possible de définir la version symétrique de cet algorithme de distance d'édition, par un simple jeu de miroir, en décomposant les forêts *par la gauche*, au lieu de les décomposer *par la droite*. L'équation 3.2 est remplacée par l'équation 3.3

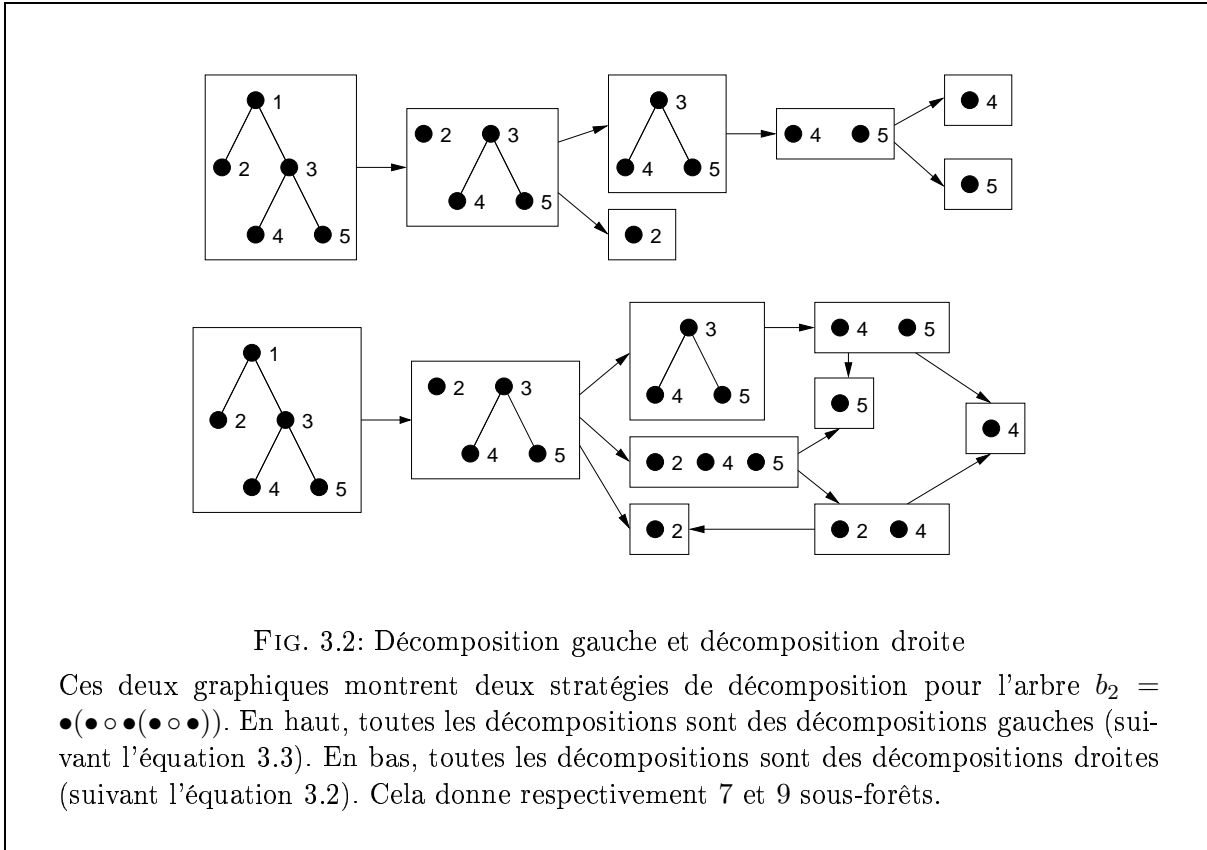
$$d(\ell(f) \circ t, \ell'(g) \circ v) = \min \begin{cases} c_d(\ell) + d(f \circ t, \ell'(g) \circ v) & \text{délétion de } \ell \\ c_i(\ell') + d(\ell(f) \circ t, g \circ v) & \text{insertion de } \ell' \\ d(\ell(f), \ell'(g)) + d(t, v) & \text{substitution de } \ell \text{ en } \ell' \end{cases} \quad (3.3)$$

Ce changement conduit naturellement à un algorithme dont les complexités en moyenne et dans le pire des cas sont identiques à celles de l'algorithme initial. Mais ce n'est plus vrai quand on regarde au cas par cas, pour un couple d'arbres donné. Le nombre total de sous-forêts induites par l'équation 3.2 ou l'équation 3.3 dépend de la morphologie de l'arbre. Cette remarque anodine va motiver toute la suite de cette section. Par exemple, pour la famille des arbres peignes, chaque arbre b_n génère $3n + 1$ sous-forêts droites, au lieu de $2n^2 + 1$. La figure 3.2 énumère toutes les forêts pour b_2 suivant que l'on applique une stratégie de décomposition droite ou gauche. Sur les arbres peignes, la complexité passe ainsi de $O(n^4)$ à $O(n^2)$.

3.2.3 Algorithme de Klein

L'algorithme proposé par Klein [77] apporte deux nouvelles idées : alterner la direction de décomposition, droite ou gauche, dans le schéma de programmation dynamique, et recourir à une décomposition suivant les *chemins lourds*. La notion de chemins lourds a été introduite par Harel et Tarjan [53].

Définition 9 (chemins lourds [53]). *Soit A un arbre enraciné. Pour chaque nœud i de A , on définit le poids de i comme la taille de $A(i)$. Pour chaque nœud interne i , $heavy(i)$ le fils de i de poids maximal. La séquence de nœuds $i, heavy(i), heavy(heavy(i)), \dots$ définit un chemin descendant, que l'on appelle un chemin lourd.*



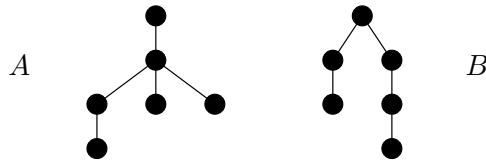
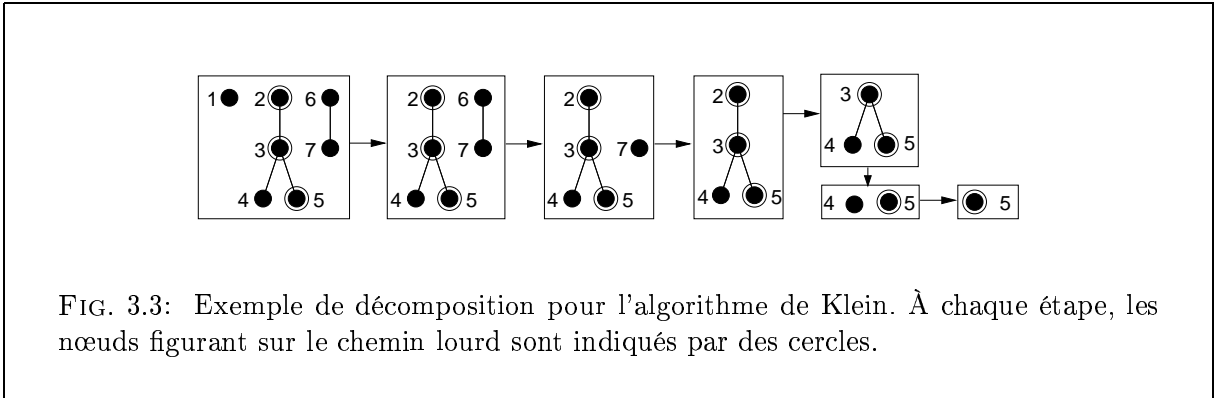
Les chemins lourds servent de guide de décomposition pour l'arbre directeur.

1. soit $(\ell(f) \circ t, G)$ le couple de forêts à comparer,
2. calculer un chemin lourd P pour $\ell(f) \circ t$,
3. si ℓ appartient à P , appliquer l'équation 3.2, sinon appliquer l'équation 3.3,
4. appliquer cette approche récursivement à toutes les sous-forêts de $\ell(f) \circ t$.

La figure 3.3 montre un exemple de décomposition pour un arbre. La contrepartie des chemins lourds dans l'algorithme de Zhang-Shasha sont en quelque sorte les chemins gauches, qui descendraient à travers les fils gauches. La décomposition en chemins lourds présente un avantage compétitif en terme de nombre de sous-forêts pour l'arbre directeur. Elle génère au plus $O(n \log(n))$ forêts, comparé à $O(n^2)$ forêts gauches ou $O(n^2)$ forêts droites dans l'algorithme de Zhang-Shasha. Le nombre de sous-forêts dans l'autre arbre reste quant à lui en $O(n^2)$.

Proposition 3 (Klein [77]). *On peut calculer la distance d'édition entre deux arbres en temps $O(n^3 \log(n))$.*

La proposition 3 n'implique toutefois pas que l'approche de Klein soit toujours plus efficace que celle de Zhang-Shasha. Cela dépend de la forme des arbres à comparer. Le couple d'arbres A et B donne un exemple où l'algorithme de Zhang-Shasha est plus satisfaisant que celui de Klein : il nécessite 72 appels récursifs distincts, contre 84 pour Klein.



L'explication vient du caractère asymétrique de la construction de Klein. L'optimisation de la décomposition en chemins lourds porte uniquement sur l'arbre directeur. Le prix à payer est que cette décomposition peut induire un nombre de sous-forêts supérieur pour le second arbre.

3.2.4 Analyse des stratégies de décomposition

Pour faire une analyse formelle des équations 3.2 et 3.3, nous avons introduit le cadre des *stratégies de décomposition*. Ce point de vue permet une présentation unifiée, tout en conduisant à un nouvel algorithme de calcul de distance.

Définition 10 (Stratégie). Soient F et G deux forêts. Une stratégie est une application de $\text{Special}(F) \times \text{Special}(G)$ dans $\{\text{left}, \text{right}\}$.

Chaque stratégie génère un ensemble spécifique d'appels récursifs sur des couples de sous-forêts. Nous les appelons des forêts *pertinentes*. Cette terminologie a été introduite par [77], où elle est utilisée pour l'arbre directeur. Nous l'étendons aux couples de forêts.

Définition 11 (Forêts pertinentes). Soit (F, G) , un couple de forêts et un algorithme de programmation dynamique qui travaille sur des sous-forêts. L'ensemble $\text{Pertinente}(F, G)$ des forêts pertinentes est le plus petit sous-ensemble de $\text{Special}(F) \times \text{Special}(G)$ tel que si la décomposition de (F, G) comprend (F', G') , alors (F', G') appartient à $\text{Pertinente}(F, G)$.

La taille de $\text{Pertinente}(F, G)$ donne une mesure de complexité pour l'algorithme de calcul de distance associé. Il est possible d'analyser l'algorithme de Zhang-Shasha et l'algorithme de Klein en les plongeant dans le cadre plus général des *stratégies par recouvrement*.

Définition 12 (Recouvrement). Soit F une forêt. Un recouvrement r de F est une application de F dans $F \cup \{\text{right}, \text{left}\}$ telle que tout nœud i de F satisfasse

- si $\text{deg}(i) = 0$ ou $\text{deg}(i) = 1$, alors $r(i) \in \{\text{right}, \text{left}\}$,
- si $\text{deg}(i) > 1$, alors $r(i)$ est un fils de i .

Dans le premier cas, $r(i)$ est appelé la direction de i , et dans le second cas, $r(i)$ est appelé le fils favori de i .

À un couple d'arbres (A, B) et un recouvrement r pour A , on peut associer de manière unique une stratégie de décomposition, de la manière suivante :

- si $\text{deg}(i) = 0$ ou $\text{deg}(i) = 1$, alors $\phi(A(i), G) = r(i)$, pour toute forêt G de B .
- si $A(i)$ est de la forme $l(A_1 \circ \dots \circ A_n)$ avec $n > 1$, alors soit $p \in \{1, \dots, n\}$ tel que le fils favori de i soit la racine de A_p . Pour toute forêt G de B , on définit

$$\begin{aligned} \phi(A(i), G) &= \text{right si } p = 1, \text{ left sinon,} \\ \phi(T \circ A_p \circ \dots \circ A_n, G) &= \text{left, pour toute forêt } T \text{ de } A_1 \circ \dots \circ A_{p-1}, \\ \phi(A_p \circ T, G) &= \text{right, pour toute forêt } T \text{ de } A_{p+1} \circ \dots \circ A_n. \end{aligned}$$

Les algorithmes de Zhang-Shasha et Klein sont des cas particuliers de ce schéma de définition. L'intérêt des recouvrements est qu'il est possible de calculer de manière exacte le nombre de forêts pertinentes associées.

Théorème 11 (Dulucq et Touzet [30]). *Soit (A, B) un couple d'arbres, A étant muni d'un recouvrement. Pour tout nœud i de A , on définit l'ensemble $E(i)$ par*

$$E(i) = \{g \in \text{Special}(B), (A(i), g) \in \text{Pertinente}(A, B)\}.$$

Il y a exactement trois possibilités pour $E(i)$:

- $E(i)$ est égal à $\text{Droite}(B)$,
- $E(i)$ est égal à $\text{Gauche}(B)$,
- $E(i)$ est égal à $\text{Special}(B)$.

On a de plus les formules de récurrence suivantes, où $\text{Free}(A)$ donne le nombre total de forêts pertinentes.

1. Si A est réduit à un unique nœud, dont la direction est droite

$$\begin{aligned} \text{Free}(A) &= \text{Left}(A) = \#\text{left}(B) \\ \text{All}(A) &= \text{Right}(A) = \#\text{special}(B) \end{aligned}$$

2. Si A est réduit à un unique nœud, dont la direction est gauche

$$\begin{aligned} \text{Free}(A) &= \text{Right}(A) = \#\text{right}(B) \\ \text{All}(A) &= \text{Left}(A) = \#\text{special}(B) \end{aligned}$$

3. Si $A = l(A')$ et si la direction de l est droite

$$\begin{aligned} \text{Free}(A) &= \text{Left}(A) = \#\text{left}(B) + \text{Right}(A') \\ \text{All}(A) &= \text{Right}(A) = \#\text{special}(B) + \text{All}(A') \end{aligned}$$

4. Si $A = l(A')$ et si la direction de l est gauche

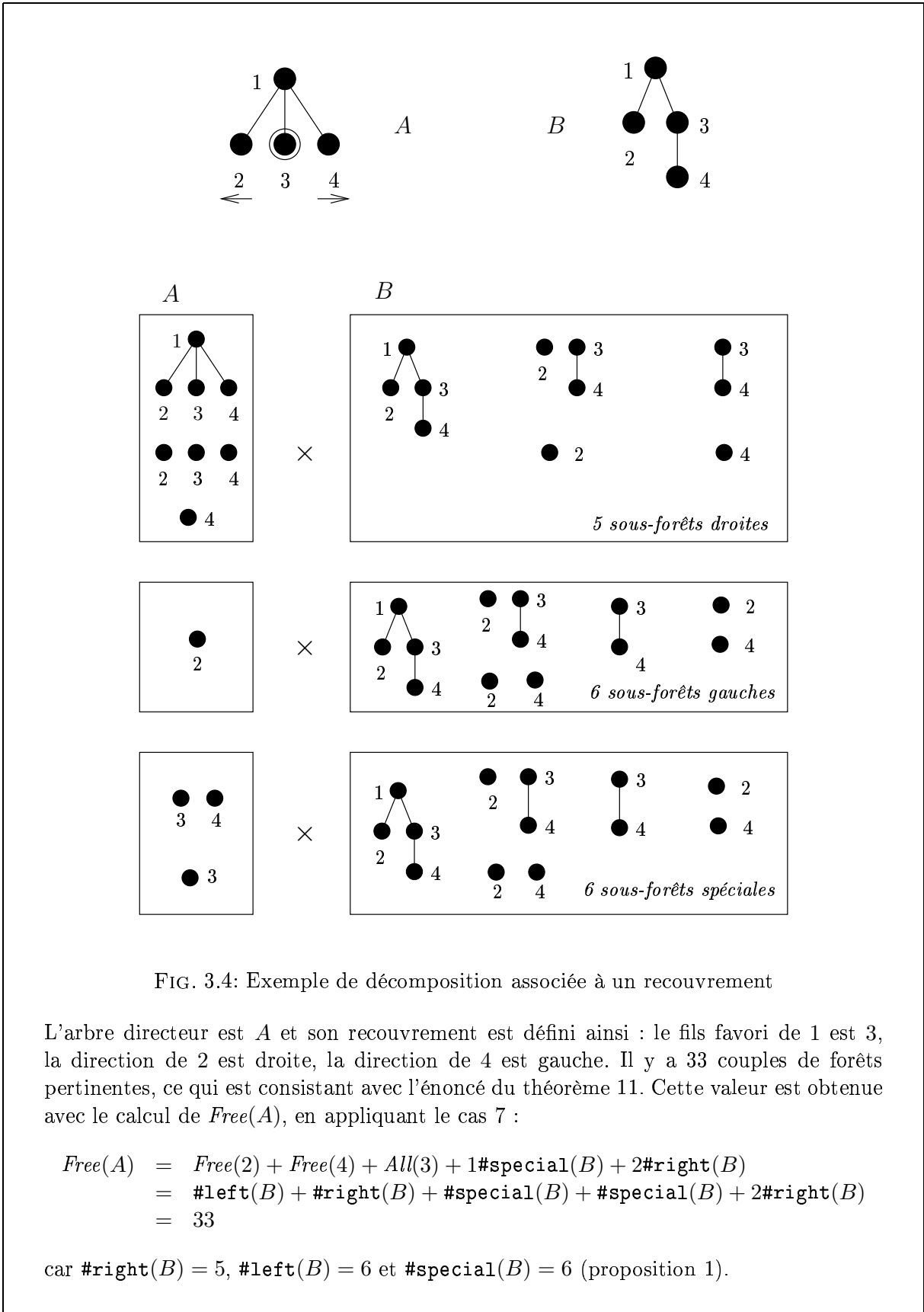
$$\begin{aligned} \text{Free}(A) &= \text{Right}(A) = \#\text{right}(B) + \text{Left}(A') \\ \text{All}(A) &= \text{Left}(A) = \#\text{special}(B) + \text{All}(A') \end{aligned}$$

5. Si $A = l(A_1 \circ \dots \circ A_n)$ et le fils favori est le fils gauche

$$\begin{aligned} \text{Free}(A) &= \text{Left}(A) = \sum_{i>1} \text{Free}(A_i) + \text{Left}(A_1) + \#\text{left}(B)(|A| - |A_1|) \\ \text{All}(A) &= \text{Right}(A) = \sum_{i>1} \text{Free}(A_i) + \text{All}(A_1) + \#\text{special}(B)(|A| - |A_1|) \end{aligned}$$

6. Si $A = l(A_1 \circ \dots \circ A_n)$ et le fils favori est le fils droit

$$\begin{aligned} \text{Free}(A) &= \text{Right}(A) = \sum_{i<n} \text{Free}(A_i) + \text{Right}(A_n) + \#\text{right}(B)(|A| - |A_n|) \\ \text{All}(A) &= \text{Left}(A) = \sum_{i<n} \text{Free}(A_i) + \text{All}(A_n) + \#\text{special}(B)(|A| - |A_n|) \end{aligned}$$



7. Sinon, $A = l(A_1 \circ \dots \circ A_n)$ et le fils favori est A_j avec $1 < j < n$

$$\text{Free}(A) = \sum_{i \neq j} \text{Free}(A_i) + \text{All}(A_j) + \#\text{right}(B)(1 + |A_1 \circ \dots \circ A_{j-1}|) + \#\text{special}(B)|A_{j+1} \circ \dots \circ A_n|$$

$$\text{Right}(A) = \text{Free}(A)$$

$$\text{All}(A) = \text{Left}(A) = \sum_{i \neq j} \text{Free}(A_i) + \text{All}(A_j) + \#\text{special}(B)(|A| - |A_j|)$$

Que dit ce théorème? Il permet de savoir *a priori*, pour un couple d'arbres donné, lequel de l'algorithme de Zhang-Shasha ou Klein est le plus efficace. La Figure 3.4 l'illustre en détaillant l'ensemble des forêts pertinentes pour un couple d'arbres muni d'un recouvrement.

Ce n'est pas le seul intérêt. Cette analyse permet de définir un algorithme efficace pour construire un recouvrement optimal, c'est-à-dire qui minimise le nombre de couples de forêts pertinentes. L'algorithme procède ainsi : pour chaque couple d'arbres (A, B) , quatre tables de programmation dynamique de taille $|A|$ sont définies, **Right**, **Left**, **Free** et **All**. La définition de **Right**, **Left**, **Free**, et **All** est directement empruntée au théorème 11. La seule différence est que le fils favori n'est pas donné a priori, mais est choisi à chaque étape de manière à minimiser le nombre de forêts pertinentes. Par exemple, si $A = l(A_1 \circ \dots \circ A_n)$ alors

$$\text{Free}(A) = \sum_{i \geq 1} \text{Free}(A_i) + \min \begin{cases} \text{Left}(A_1) - \text{Free}(A_1) + \#\text{left}(B)(|A| - |A_1|) \\ \text{All}(A_j) - \text{Free}(A_j) + \#\text{special}(B)|A_{j+1} \circ \dots \circ A_n| \\ \quad + \#\text{right}(B)(1 + |A_1 \circ \dots \circ A_{j-1}|), \quad 1 < j < n \\ \text{Right}(A_n) - \text{Free}(A_n) + \#\text{right}(B)(|A| - |A_n|) \end{cases}$$

Le fils favori est sélectionné comme la racine du sous-arbre A_j qui réalise la valeur maximale dans l'alternative. Le recouvrement optimal est ensuite construit par chaînage arrière à partir de **Free**(A). Le coût total de ce prétraitement est linéaire. Par construction, le nombre de forêts pertinentes est inférieur au nombre de forêts générées par les algorithmes de Zhang-Shasha ou de Klein. Il est donc au plus en $O(n^3 \log(n))$ dans le pire des cas et en $O(n^3)$ en moyenne.

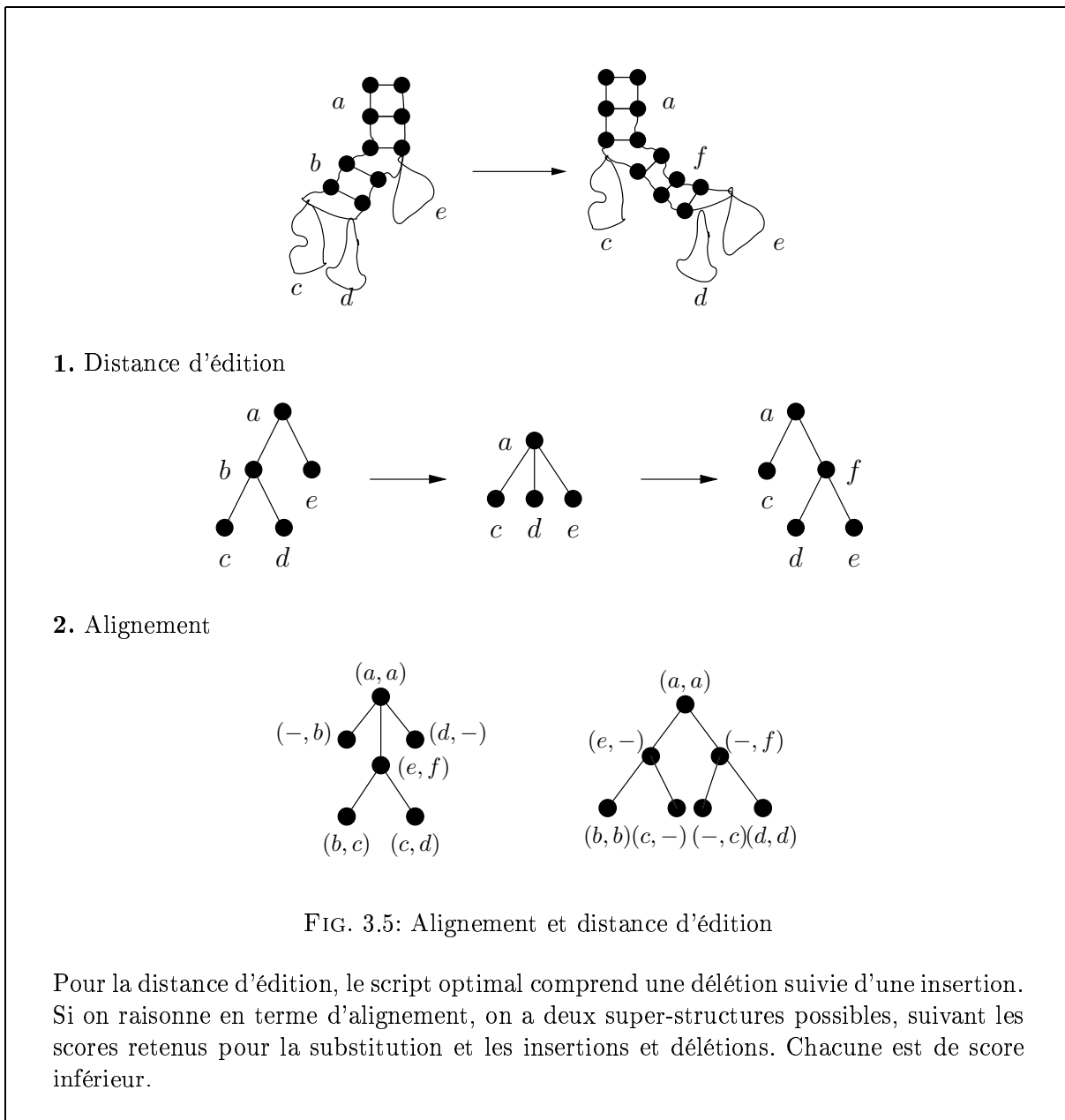
3.3 Alignement d'arbres

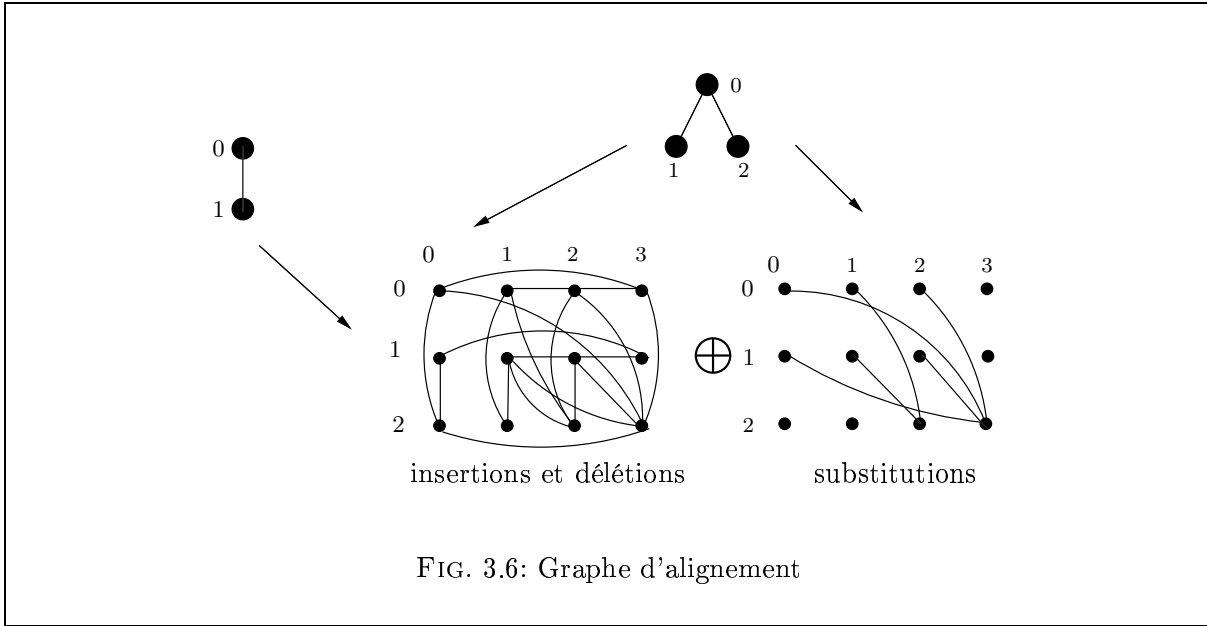
3.3.1 Alignement versus distance

En 1995, Jiang *et al.* [70] ont introduit une nouvelle approche pour la comparaison de structures secondaires en appliquant à la lettre la notion d'alignement telle qu'elle est décrite dans les manuels de bio-informatique pour l'ADN ou les protéines, c'est-à-dire en « ajoutant des espaces ».

Définition 13. Soient A et B deux arbres étiquetés dans \mathcal{L} . Un alignement entre A et B est un arbre C étiqueté dans $(\mathcal{L} \cup \{-\})^2$ tel que la projection de C sur la première coordonnée soit égale à A , et sur la deuxième coordonnée soit égale à B . Un alignement définit un script d'édition : chaque nœud de $\mathcal{L} \times \mathcal{L}$ est une substitution, de $\{-\} \times \mathcal{L}$ une insertion et de $\mathcal{L} \times \{-\}$ une délétion. L'alignement optimal est l'alignement dont le script d'édition sous-jacent est de score maximal.

Sur les séquences, les définitions de distance et d'alignement coïncident. Ce n'est plus le cas en général sur les arbres. La Figure 3.5 montre un exemple typique où l'alignement produit un script de score inférieur à celui de la distance complète. Il n'y a pas de ratio de garantie constant entre le score de l'alignement et celui de la distance. Soient $A = \bullet^n \circ \bullet(\bullet^n \circ \bullet^n)$ et $B = \bullet(\bullet^n \circ \bullet^n) \circ \bullet^n$. La distance entre A et B est 2, une délétion suivie d'une insertion. L'alignement entre A et B conduit à un mapping dont le coût est en $O(n)$, avec au moins $n - 1$ délétions.





L'alignement définit une super-structure, alors que la distance définit un sous-arbre commun. Du point de vue du script d'édition, cela revient à imposer que toutes les insertions aient lieu avant les délétions, ce qui autorise moins de remaniements internes. Comment cela se traduit-il sur le graphe d'édition ? Si un nœud i est détruit, $A(i)$ est nécessairement associé avec une sous-forêt close de B : tout chemin qui emprunte un arc horizontal $(i, j) \rightsquigarrow (i + 1, j)$ doit ensuite passer par un nœud $(i + |A(i)|, j + |f|)$ où f est une forêt close, éventuellement vide, qui débute en j . Les arcs correspondant à des délétions ou des insertions se trouvent donc modifiés de la sorte :

- l'arc $(i, j) \rightsquigarrow (i + 1, j)$ est remplacé par l'ensemble des arcs $\{(i, j) \rightsquigarrow (i + |A(i)|, j + |f|) \mid f \text{ est une sous-forêt close de } B \text{ qui débute en } j\}$
- l'arc $(i, j) \rightsquigarrow (i, j + 1)$ est remplacé par l'ensemble des arcs $\{(i, j) \rightsquigarrow (i + |f|, j + |B(j)|) \mid f \text{ est une sous-forêt close de } A \text{ qui débute en } i\}$

Les arcs associés à des substitutions ne changent pas. On obtient ainsi un multigraphe, que nous appelons le *graphe d'alignement*. La figure 3.6 donne le graphe d'alignement pour le couple d'arbres de la figure 3.1.

3.3.2 Algorithme de Jiang pour l'alignement

L'alignement optimal est construit par programmation dynamique, en étant guidé par le super-arbre à construire.

$$\text{Align}(\ell(f), \ell'(f')) = \max \begin{cases} c_s(\ell, \ell') + \text{Align}(f, f') & \text{substitution} \\ c_i(\ell') + \max\{\text{Align}(u, f') + \text{Align}(v, \varepsilon) \mid u \circ v = \ell(f)\} & \text{insertion} \\ c_d(\ell) + \max\{\text{Align}(f, u') + \text{Align}(\varepsilon, v') \mid u' \circ v' = \ell'(f')\} & \text{délétion} \end{cases} \quad (3.4)$$

$$\text{Align}(\ell(f) \circ t, \ell'(f') \circ t') = \max \begin{cases} \text{Align}(\ell(f), \ell'(f')) + \text{Align}(t, t') & \text{substitution} \\ c_i(\ell') + \max\{\text{Align}(u, f') + \text{Align}(v, t') \mid u \circ v = \ell(f) \circ t\} & \text{insertion} \\ c_d(\ell) + \max\{\text{Align}(f, u') + \text{Align}(t, v') \mid u' \circ v' = \ell'(f') \circ t'\} & \text{délétion} \end{cases} \quad (3.5)$$

La substitution crée un nœud (ℓ, ℓ') dont les descendants sont $\text{Align}(f, f')$, l'insertion un nœud $(-, \ell')$ dont les descendants sont $\text{Align}(u, f')$ et la délétion un nœud $(\ell, -)$ dont les descendants sont $\text{Align}(f, u')$.

L'ensemble des couples de forêts pertinentes qui apparaissent dans cette décomposition est exactement $\text{Suffixe}(A) \times \text{Close}(B) \cup \text{Close}(A) \times \text{Suffixe}(B)$. Le nombre de forêts pertinentes est ainsi majoré par $(|A| - 1) \times (|B| - 1) \cdot \max\{\text{deg}(A) + \text{deg}(B)\}$. La complexité en temps est en $O(|A| \cdot |B| \cdot (\text{deg}(A) + \text{deg}(B))^2)$. On obtient donc une complexité asymptotique en $O(n^4)$, ce qui est théoriquement moins bon que le $O(n^3 \log(n))$ de la distance d'édition. La complexité en moyenne n'est pas connue. Toutefois, dans le cadre des structures d'ARN, avec les représentations compactes, le degré pour les structures connues est majoré par 4. L'attrait de l'alignement par rapport à la distance d'édition complète est qu'il pourrait conduire à un algorithme plus efficace en pratique, tout en restant sensible.

3.4 Prise en compte des gaps

Le schéma de score utilisé jusqu'ici (sections 3.2 et 3.3) considère que les événements d'insertion ou de délétion sont indépendants les uns des autres, ce qui est simplificateur. La formation de trois délétions isolées, par exemple, est un événement évolutif moins probable que trois délétions successives qui résulteraient d'un même événement mutationnel. La solution classique est de gérer l'insertion ou la délétion sous forme de *gaps* et de choisir un système de score qui favorise la formation de gaps longs et peu nombreux, au détriment de nombreux gaps brefs. On introduit pour cela des fonctions de pénalité non additives, avec des coûts affines (qui supposent une dépendance d'ordre 1) ou convexes. La prise en compte des gaps est particulièrement sensible pour les représentations des structures secondaires ponctuelles, base par base.

3.4.1 Le problème de la définition

Dans une séquence d'ADN ou protéique, un *gap* est une suite contigue de nucléotides ou d'acides aminés : c'est un facteur. Il y a plusieurs manières d'adapter ce point de vue aux structures d'ARN. La première est de considérer qu'un gap dans un arbre est un gap dans la séquence d'ARN associée. C'est la définition qui est appliquée dans [142]. Nous considérons ici une définition plus générale, où une insertion ou une délétion se propage de 5' en 3', mais également à ses fils : un gap est un sous-arbre quelconque.

3.4.2 Gaps affines

Le système de pénalités pour les gaps affines est décrit par deux paramètres : la pénalité d'ouverture Ouv et celle d'extension Ext .

Pour la distance d'édition, la dépendance entre un père et son fils pour les pénalités de gaps fait que le schéma de récurrence des équations 3.2 ou 3.3 doit être modifié. On définit $s(i, j)$ le score du mapping optimal entre $A(i)$ et $B(j)$ qui substitue i en j . La donnée de s résout le problème général de distance, en ajoutant une racine virtuelle à chacun des arbres. Pour tout

couple (i, j) de $A \times B$, $s(i, j)$ est calculé à partir des valeurs de $s(k, l)$ pour tout l descendant de i et tout k de j . On reprend la topologie du graphe d'édition, limité aux nœuds de $A(i) \times B(j)$, en modifiant l'étiquetage des arcs.

- le label des arcs délétions et insertions dépend de la position de (k, l) par rapport à (i, j) :

$$\begin{aligned} (k, l) \rightsquigarrow (k+1, l) & \text{ est étiqueté par } \mathbf{Ouv} + \mathbf{Ext} \text{ si } k \text{ est un fils de } i, \mathbf{Ext} \text{ sinon,} \\ (k, l) \rightsquigarrow (k, l+1) & \text{ est étiqueté par } \mathbf{Ouv} + \mathbf{Ext} \text{ si } l \text{ est un fils de } j, \mathbf{Ext} \text{ sinon.} \end{aligned}$$

- le coût d'un arc substitution issu de (k, l) est $s(k, l)$.

$s(i, j)$ est égal au score du chemin optimal entre $(i+1, j+1)$ et $(i+|A(i)|-1, j+|B(j)|-1)$ additionné de $c_s(i, j)$. Cette procédure mène à un algorithme en $O(n^4)$.

Le modèle de l'alignement se prête mieux à l'introduction des gaps. L'implémentation utilise, à l'image des algorithmes classiques pour l'alignement de séquences, trois matrices : a pour les mappings qui terminent par une substitution, b par une délétion et c par une insertion. Les couples de forêts pertinentes sont identiques à ceux des 3.4 et 3.5, ce qui fait que l'on conserve la même complexité.

$$\begin{aligned} a(\ell(f) \circ t, \ell'(f') \circ t') = \\ \max \left\{ \begin{array}{ll} c_s(\ell, \ell') + a(f, f') + a(t, t') & \text{substitution de } \ell \text{ en } \ell' \\ \mathbf{Ouv} + \mathbf{Ext} + \max\{c(u, f') + a(v, t') \mid u \circ v = \ell(f) \circ t\} & \text{insertion de } \ell' \\ \mathbf{Ouv} + \mathbf{Ext} + \max\{b(f, u') + a(t, v') \mid u' \circ v' = \ell'(f') \circ t'\} & \text{délétion de } \ell \end{array} \right. \end{aligned}$$

$$\begin{aligned} b(\ell(f) \circ t, \ell'(f') \circ t') = \\ \max \left\{ \begin{array}{ll} c_s(\ell, \ell') + a(f, f') + b(t, t') & \text{substitution de } \ell \text{ en } \ell' \\ \mathbf{Ouv} + \mathbf{Ext} + \max\{c(u, f') + b(v, t') \mid u \circ v = \ell(f) \circ t\} & \text{insertion de } \ell' \\ \mathbf{Ext} + \max\{b(f, u') + b(t, v') \mid u' \circ v' = \ell'(f') \circ t'\} & \text{délétion de } \ell \end{array} \right. \end{aligned}$$

$$\begin{aligned} c(\ell(f) \circ t, \ell'(f') \circ t') = \\ \max \left\{ \begin{array}{ll} c_s(\ell, \ell') + a(f, f') + c(t, t') & \text{substitution de } \ell \text{ en } \ell' \\ \mathbf{Ext} + \max\{c(u, f') + c(v, t') \mid u \circ v = \ell(f) \circ t\} & \text{insertion de } \ell' \\ \mathbf{Ouv} + \mathbf{Ext} + \max\{b(f, u') + c(t, v') \mid u' \circ v' = \ell'(f') \circ t'\} & \text{délétion de } \ell \end{array} \right. \end{aligned}$$

L'alignement avec pénalités de gap affines peut également être présenté en terme de modèles de Markov cachés, comme dans le cas des séquences [31]. Le modèle a trois états S , I et D , correspondant à une substitution, une insertion et une délétion, qui sont la contrepartie des trois matrices a , b et c . Les transitions de S et D vers I , de S et I vers D sont labelées par le coût d'ouverture de gaps, et les transitions de D vers D et de I vers I par les pénalités d'extension. Le modèle est modifié pour émettre des couples de $\{A, C, G, U, -\}^2$, qui forment les colonnes de l'alignement deux à deux. Ce point de vue peut être étendu à l'alignement de structures d'ARN avec arité bornée en utilisant des automates d'arbres stochastiques [115].

3.4.3 Gaps convexes

Le modèle de pénalités universel est celui des gaps convexes, qui généralise les gaps affines : $\mathbf{Gap}(t_1(t_2)) \leq \mathbf{Gap}(t_1) + \mathbf{Gap}(t_2)$, où t_1 et t_2 sont des arbres et $t_1(t_2)$ est un arbre tel que la racine de t_2 est connectée à t_1 . Pour les séquences, l'alignement avec gaps convexes est réalisable en $O(n^2 \log(n))$ [41]. En passant sur les arbres, le problème change de classe de complexité.

Théorème 12 (Touzet [134]). *Les problèmes de la distance d'édition et de l'alignement sur les arbres avec pénalités de gap convexes sont NP-durs.*

La démonstration repose sur la réduction du problème $X3C$ (*exact cover by 3-sets*, [43]). On compare des arbres construits avec l'ensemble de labels $\mathcal{L} = \{\bullet\} \cup \mathbb{N}$. Les coûts de substitution sont

$$\begin{aligned} \text{Sub}(i, \bullet) &= \text{Sub}(\bullet, i) = -1, & \forall i \in \mathbb{N} \\ \text{Sub}(i, j) &= -1.5, & \forall i \in \mathbb{N}, \forall j \in \mathbb{N} \end{aligned}$$

Un arbre T satisfait la propriété (\star) , si pour tout nœud i de T , si $i \in \mathbb{N}$, alors T contient un seul nœud étiqueté par i . La fonction de coût Gap est définie par

$$\begin{aligned} \text{Gap}(t) &= -|t| - 1, & \text{si } t \text{ satisfait la propriété } (\star), \\ &= -|t| - 1.5, & \text{sinon.} \end{aligned}$$

3.4.4 Gaps avec sous-arbres complets

Au sujet des gaps, une dernière solution consiste à limiter les gaps aux sous-arbres complets. Cela revient à imposer qu'à chaque étape du script d'édition les opérations d'insertion ou de délétion ne s'appliquent qu'aux feuilles. Bien que radicale, cette distance d'édition restreinte reste expressive quand on travaille sur l'ARN. Prenant les représentations compactées où un nœud interne est toute une tige ou avec le codage SIBMH (section 2.3.4), elle autorise

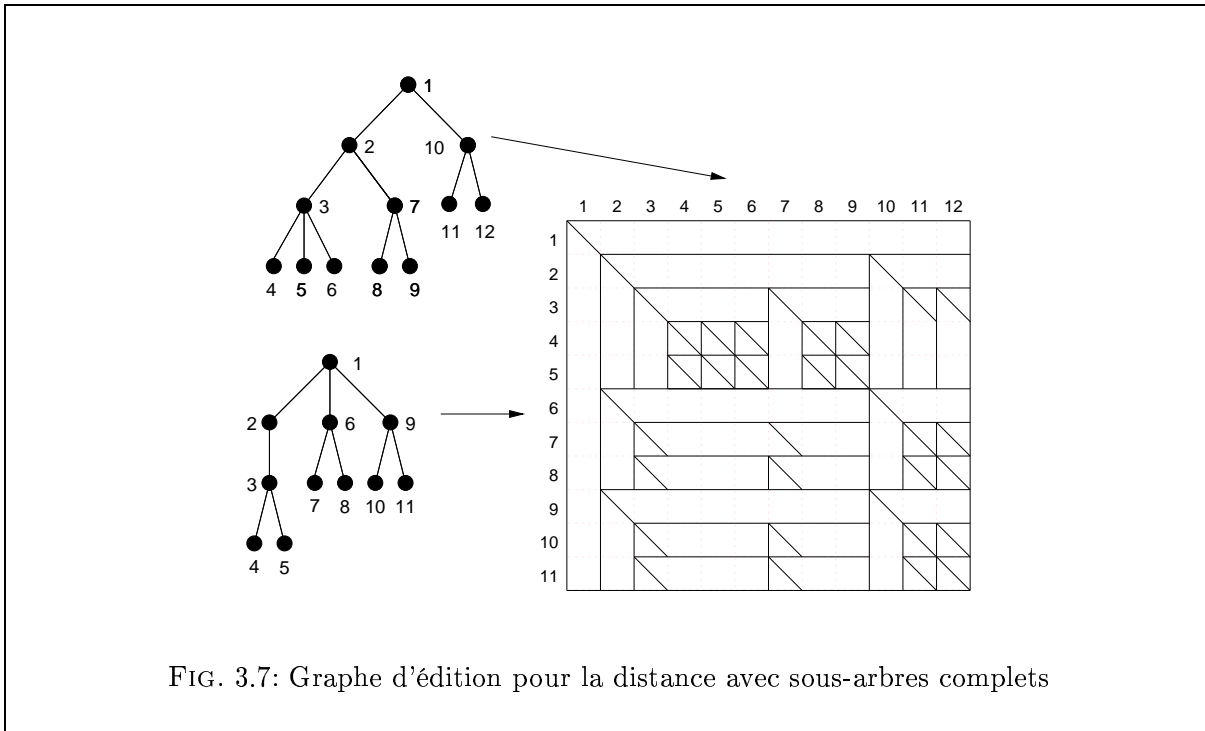
- l'insertion, la délétion ou la substitution d'une paire de bases appariées au sein d'une tige, via la substitution,
- l'insertion, la délétion ou la substitution d'une base libre au sein d'une boucle, via la substitution,
- l'insertion ou la délétion de sous-structures entières, via l'insertion ou la délétion.

Une substitution ne peut avoir lieu qu'entre nœuds de même profondeur, une délétion doit s'accompagner de la délétion de tous les fils, une insertion également. Cette restriction des opérations d'édition conduit à un graphe d'édition modifié particulièrement simple, qui peut être construit et étiqueté *a priori*, comme dans le cas des mots. La figure 3.7 illustre cela. Cette propriété est exploitée par Chawathe [15] avec des fonctions de gaps linéaires, ce qui conduit à un algorithme quadratique en espace et en temps.

Nous avons proposé un algorithme alternatif, qui permet de prendre en charge des fonctions de pénalité de gaps quelconques sans surcoût. Cette méthode est particulièrement adaptée pour l'alignement de structures dont le cœur est conservé. Son intérêt réside également dans son efficacité pour effectuer des comparaisons à grande échelle, dans le but de faire du clustering entre plusieurs structures potentielles par exemple.

Proposition 4 (Touzet [134]). *Soient $\ell, \ell' \in \mathcal{L}$, et soient f, f' deux forêts.*

$$\begin{aligned} d(\ell(f), \varepsilon) &= \text{Gap}(\ell(f)) \\ d(\varepsilon, \ell'(f')) &= \text{Gap}(\ell'(f')) \\ d(\ell(f), \ell'(f')) &= \max \begin{cases} d(\ell, \ell') + \text{Dist}(f, f') \\ \text{Gap}(\ell(f)) + \text{Gap}(\ell'(f')) \end{cases} \end{aligned}$$



Dist est la distance d'édition entre mots, les mots étant ici des forêts, c'est-à-dire des suites d'arbres. Cette proposition conduit à un algorithme en temps et en espace égal à

$$\sum_{\substack{i \in a, j \in b \\ \text{depth}(i) = \text{depth}(j)}} \text{deg}(i)\text{deg}(j).$$

Si on souhaite effectuer le seul calcul de la distance, sans construire de mapping, on peut améliorer la gestion de l'espace mémoire et obtenir une complexité spatiale meilleure dans tous les cas, et en $O(\text{deg}(A) \times \text{deg}(B) \times (\log(|A|) + \log(|B|)))$ dans le pire des cas.

3.5 Comparaison locale

Le problème de la comparaison locale est de trouver la région de plus forte similitude entre deux arbres A et B . Comme pour le traitement des gaps, c'est un problème à plusieurs facettes, suivant le type de sous-structure conservée recherchée. On peut considérer trois formes de régions :

1. *sous-arbre complet* : quel est le couple (i, j) de $A \times B$ tel que la similitude entre $A(i)$ et $B(j)$ soit de score optimal,
2. *forêt close* : quel est le couple (f, g) de $\text{Close}(A) \times \text{Close}(B)$ de similitude maximale
3. *sous-arbre quelconque* : quel est le couple (f, g) de sous-arbres de $A \times B$ de similitude maximale.

Ces différentes interprétations répondent en fait à des objectifs distincts. Dans les deux premiers cas, on cherche à identifier les sous-structures terminales communes. Cette formulation est particulièrement pertinente quand il s'agit de trouver des motifs structuraux communs entre deux

structures, exactes ou prédites (voir figure 3.8). La dernière définition, quant à elle, permet de comparer deux structures homologues ayant fortement divergé peut-être plus spécifiquement au niveau des tiges terminales. C'est le pendant des gaps avec des sous-arbres complets, limités aux sous-structures terminales, de la section 3.4.4 .

3.5.1 Rechercher les sous-arbres complets optimaux

Que ce soit pour la distance ou l'alignement, les descriptions des sections 3.2 et 3.3 montrent que les couples de sous-arbres complets figurent parmi les forêts pertinentes. Le couple de sous-arbres complets de score optimal est donc obtenu directement comme un sous-produit de la décomposition par programmation dynamique pour la comparaison globale.

3.5.2 Rechercher les forêts closes optimales

La définition de l'alignement local sur les forêts closes est celle retenue dans [56], et mise en œuvre dans le logiciel *RNAforester*. La comparaison locale y est définie à partir de l'algorithme d'alignement de la section 3.3. Comme l'ensemble des forêts pertinentes ne comprend pas $Close(A) \times Close(B)$, la solution proposée est d'ajouter explicitement le calcul de $\mathbf{Align}(f, g)$ pour tout couple (f, g) de forêts closes aux équations 3.4 et 3.5 pour l'alignement global. On obtient ainsi un algorithme d'alignement local en temps $O(|A| \cdot |B| \cdot (deg(A) + deg(B))^3)$, au lieu de $O(|A| \cdot |B| \cdot (deg(A) + deg(B))^2)$ pour l'alignement global.

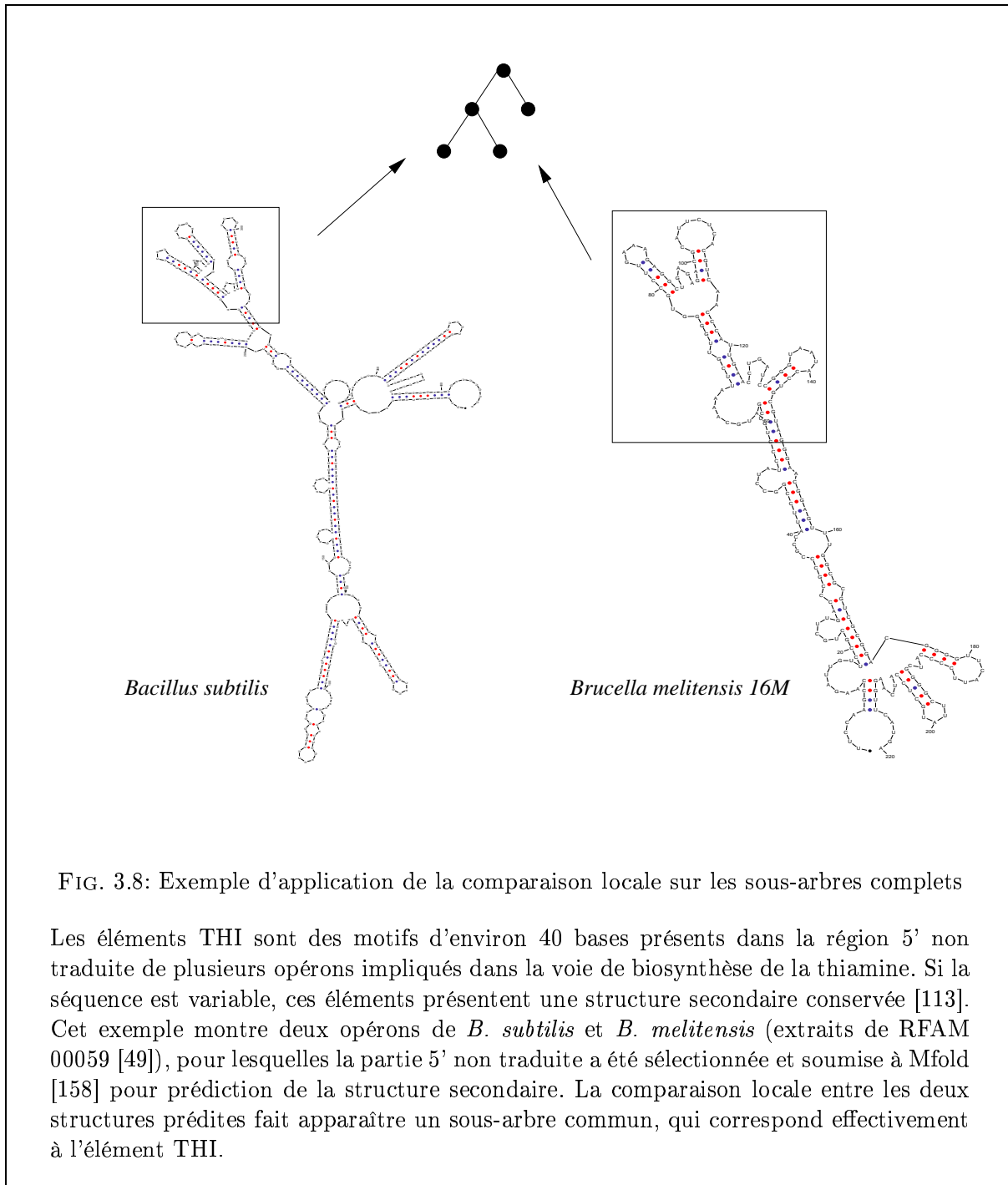
Nous montrons que l'on peut améliorer cet algorithme, et retrouver le comportement asymptotique de l'alignement global en introduisant un second schéma de récurrence à coté de \mathbf{Align} . $B(f, g)$ est le score du meilleur alignement (global) entre f et un préfixe de g .

$$\begin{aligned}
 B(f, \varepsilon) &= \mathbf{Align}(f, \varepsilon) \\
 B(\varepsilon, g) &= 0 \\
 B(\ell(f) \circ t, \ell'(f') \circ t') &= \\
 \max \left\{ \begin{array}{ll}
 c_s(\ell, \ell') + \mathbf{Align}(f, f') + B(t, t') & \text{substitution} \\
 c_i(\ell') + \max\{\mathbf{Align}(u, f') + B(v, t') \mid u \circ v = \ell(f) \circ t\} & \text{insertion} \\
 c_d(\ell) + \max\{\mathbf{Align}(f, u') + B(t, v') \mid u' \circ v' = \ell'(f') \circ t'\} & \text{délétion}
 \end{array} \right.
 \end{aligned}$$

B est ainsi calculé à partir de \mathbf{Align} pour tout couple de $Close(A) \times Suffixe(B)$. Les forêts pertinentes de ce schéma sont les mêmes que pour l'alignement global. Cette technique peut également s'appliquer à la distance d'édition.

3.5.3 Rechercher les sous-arbres optimaux

L'approche basée sur les sous-arbres quelconques est la plus générale : elle autorise des variations au niveau des structures terminales, et met en évidence la conservation de la structure du cœur de la molécule. C'est celle qui est retenue dans [35], avec la distance comme mode de comparaison. D'un point de vue algorithmique, la comparaison locale se déduit de la comparaison globale sur le même modèle que la comparaison de séquences : on néglige la contribution des régions dissimilaires, de score négatif, en mettant le score à 0. Le couple optimal est déterminé en prenant $\max(d(A(i), B(j)))$. Le même raisonnement est applicable à la distance avec les stratégies de décomposition définies à partir d'un recouvrement. Il est également valable pour l'alignement.



3.6 Intégration de la structure primaire et de la structure tertiaire

Nous avons présenté différents algorithmes de comparaison en escamotant totalement une question sensible : le choix du système de score pour les substitutions. La section 3.4 abordait seulement le problème du coût des insertions et des délétions.

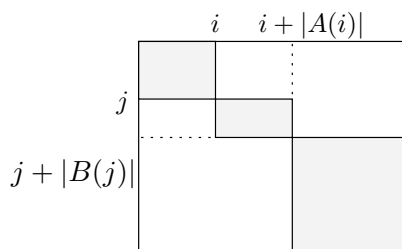
Par manque de données et de modèles, la plupart des programmes de comparaison utilise un système de score basique : un coût unique pour toutes les substitutions, un coût unique pour toutes les identités. On peut imaginer d'incorporer des ingrédients supplémentaires comme la longueur ou l'énergie des tiges, voire des informations sur la structure tri-dimensionnelle utilisée pour la modélisation [94]. Il est possible également de prendre en compte finement la structure primaire. C'est ce qui est proposé dans [78] avec les matrices de substitution RIBOSUM pour les paires de bases et pour les bases libres. La construction suit un protocole proche de celui établi pour les matrices protéiques BLOSUM avec un échantillon d'apprentissage d'ARN ribosomiques. Cette analyse systématique fait apparaître que les régions libres sont en moyenne mieux conservées que les régions appariées.

Nous pensons que le traitement uniforme de la structure primaire est délicat en raison du caractère hétérogène de sa conservation et du rôle paradoxal des mutations compensatoires. Cela suggère une approche pragmatique : utiliser les conservations locales éventuelles de la structure primaire pour guider la comparaison au niveau structurel, sous forme de *contraintes*. La figure 3.9 reprend l'exemple des RNase P de la figure 2.5 de la section 2.3. L'examen de la structure primaire révèle trois contraintes. Ces contraintes peuvent ensuite être projetées sur les différents types de représentation de la structure secondaire décrits en section 2.3.

Définition 14 (Contraintes). Soient A et B deux arbres. Un ensemble de contraintes est un sous-ensemble de $A \times B$ tel qu'un élément de A ou de B apparaisse au plus une fois.

L'intégration de contraintes permet également le passage d'information de la structure tertiaire vers la structure secondaire. Le problème de la distance entre deux ARN en tenant compte des pseudo-nœuds est NP-complet [93]. Mais quand les pseudo-nœuds sont conservés, on peut résoudre le problème en pré-alignant les pseudo-nœuds qui forment alors des contraintes sur la structure secondaire résiduelle.

On modifie les algorithmes de calcul de distance et d'alignement pour introduire la gestion de contraintes et en profiter pour élaguer l'espace de recherche. Cela se formule aisément à partir du graphe d'édition. Pour tout couple (i, j) de $A \times B$ appartenant à un ensemble de contraintes, on définit $C(i, j)$ l'ensemble des couples de nœuds du graphe d'édition compatibles avec (i, j) : $C(i, j) = \{0..i\} \times \{0..j\} \cup \{i + 1..i + |A(i)|\} \times \{j + 1..j + |B(j)|\} \cup \{i + |A(i)|..m\} \times \{j + |B(j)|..n\}$.



Proposition 5. Soit (A, B) un couple d'arbres, de taille respective m et n , muni d'un ensemble de contraintes $X = \{(i_1, j_1), \dots, (i_l, j_l)\}$

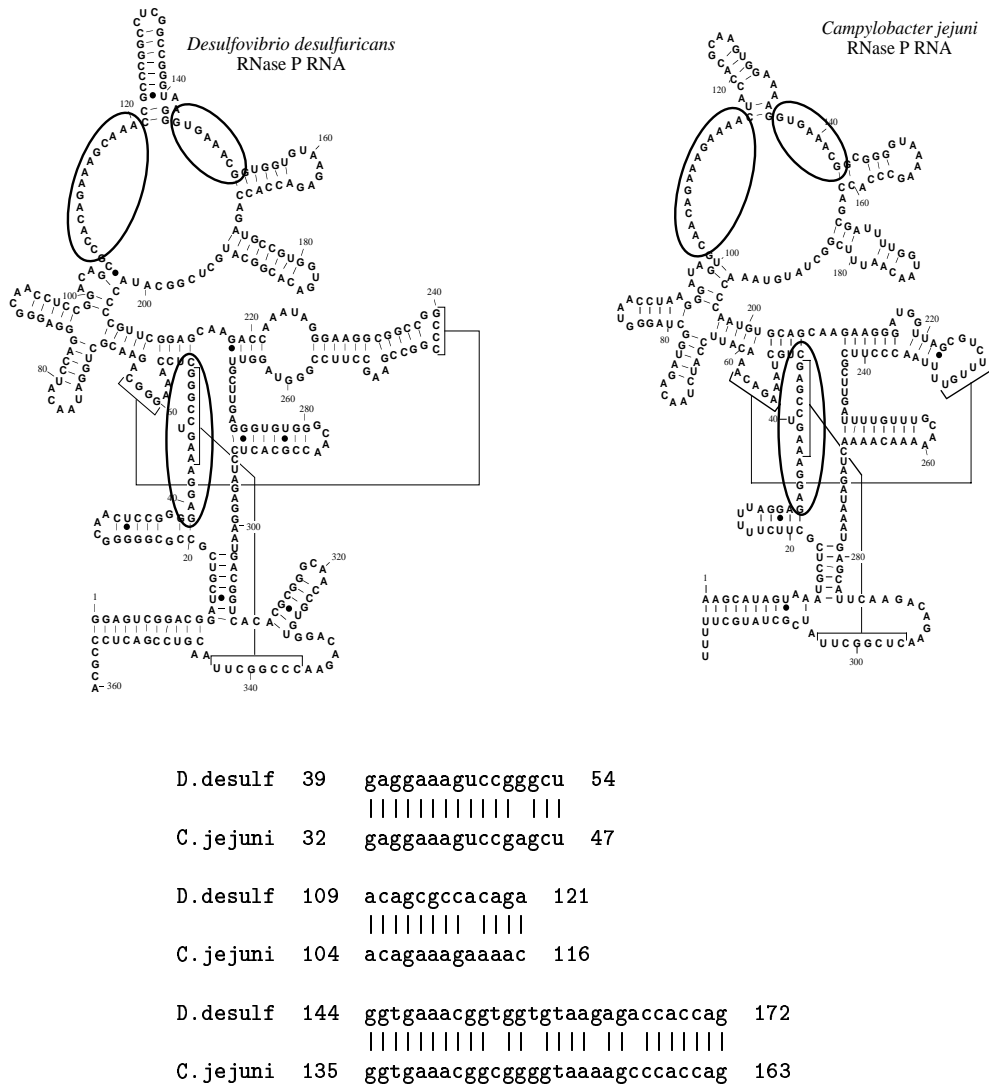


FIG. 3.9: Conservation locale de la structure primaire

Cet exemple reprend les deux structures de RNase P de la figure 2.5, qui montrait que les séquences ne s'alignaient pas correctement sur toute leur longueur. Les deux séquences présentent toutefois trois courtes régions exceptionnellement conservées, situées dans la partie 5' du premier pseudo-nœud et de part et d'autre de la multi-boucle terminale. Ces régions sont donc cohérentes avec l'alignement des structures secondaires.

- il existe un script d'édition compatible avec X s'il existe un chemin de $(0, 0)$ à (m, n) dans le graphe d'édition limité aux nœuds de $\cap_k C(i_k, j_k)$,
- le script d'édition optimal est obtenu en appliquant l'équation 3.1 aux couples de $\cap_k C(i_k, j_k)$.

Si les contraintes portent sur des boucles de l'ARN, soit sur les feuilles de l'arbre, alors il suffit qu'elles respectent l'ordre préfixe. Ce n'est plus vrai avec l'alignement. Mais on obtient une propriété analogue en s'adressant au graphe d'alignement.

Proposition 6. *Soit (A, B) un couple d'arbres, de taille respective m et n , muni d'un ensemble de contraintes $X = \{(i_1, j_1), \dots, (i_l, j_l)\}$*

- il existe un alignement compatible avec X s'il existe un chemin de $(0, 0)$ à (m, n) dans le graphe d'alignement limité aux nœuds de $\cap_k C(i_k, j_k)$,
- le script d'édition optimal est obtenu en appliquant l'équation 3.4 aux couples de $\cap_k C(i_k, j_k)$.

3.7 Perspectives

D'un point de vue algorithmique, ce chapitre laisse plusieurs questions ouvertes. Nous avons montré dans [30] qu'il existait une minoration en $O(n^2 \log^2(n))$ pour le nombre de forêts pertinentes générées par un algorithme de calcul de distance basée sur une stratégie de décomposition. Mais la question générale d'une borne inférieure, que ce soit en terme de complexité de problème ou en terme de nombres de forêts pertinentes, est ouverte pour la distance d'édition et pour l'alignement. Le problème de l'amélioration de la gestion de l'espace est également posé. Peut-on calculer une distance ou un alignement entre arbres en espace linéaire ? Cela devient-il possible si on borne le nombre d'erreurs, comme dans [66] ?

Enfin, le lecteur a dû remarquer que ces travaux passaient en revue une palette d'algorithmes sans se prononcer finalement sur la question biologique essentielle : comment comparer deux ARN ? Nous n'avons fait que la moitié du chemin : proposer sans décider. La pertinence du modèle de comparaison se décline en fait en une variété de sous-problèmes. Quel est le bon modèle de représentation ? Quel est le bon paradigme de comparaison (distance, alignement, distance restreinte) ? Quel est le bon système de score ? Quelle est la complexité en moyenne sur des « ARN aléatoires » des différents algorithmes ? Sans compter les points que nous n'avons pas abordés. Le modèle d'édition est limité ici aux trois opérations standard (insertion, délétion et substitution), mais il est possible d'adjoindre des opérations telles que la fusion de nœuds ou d'arcs [3]. Dans le cas de structures exactes, répondre à ces questions suppose de disposer d'un modèle évolutif, qui pourrait être obtenu par une analyse systématique à partir de jeux de données représentatifs. La disponibilité récente de nombreuses bases de données, telles que RFAM [49], RNABase [100], ouvre la voie à une telle étude.

Chapitre 4

Prédiction de structures secondaires

Le problème de la prédiction de structure d'ARN consiste à proposer une organisation structurale à partir de la connaissance de la seule structure primaire. Ce thème de recherche trouve sa justification dans l'écart entre la quantité de séquences disponibles et le peu de structures résolues expérimentalement.

Dans le cas d'une famille de séquences homologues, la conservation de la structure secondaire au fil de l'évolution indique que celle-ci est fonctionnellement importante. Nous nous intéressons à l'inférence de la structure dans ce contexte : les séquences partagent-elles une même structure ? Si oui, quelle est-elle ? Ce travail a été mené dans le cadre de la thèse d'Olivier Perriquet, soutenue en 2003, et a abouti à une méthode originale avec le logiciel Carnac.

4.1 Position du problème

Les premières méthodes de prédiction reposent sur le postulat que la stabilité de la structure d'un ARN est essentielle à sa fonction. Suivant les principes de la thermodynamique, l'ARN se replie ainsi dans un état d'*énergie libre minimale*. Le niveau d'énergie libre peut être approximé dans un modèle basé sur les plus proches voisins [65, 96]. La recherche de la structure secondaire optimale se formule alors comme un problème d'optimisation combinatoire qui se résout par programmation dynamique. Dans cette catégorie, les deux programmes les plus utilisés sont RNAfold [59] ou le très populaire Mfold [158, 96].

Les résultats obtenus peuvent être réellement de très bonne qualité. Par exemple, pour la séquence de l'ARN de transfert de la figure 2.3, Mfold propose comme structure

```
GGGGCUAUAGCUCAGCUGGGAGAGCGCCUGCUUUGCACGCGAGGAGGUCUGCGGUUCGAUCCCGCAUAGCUCCACCA
(((((((...(((.....))))).((((.....))))). . . . . (((.....))))).)))))) . . . .
```

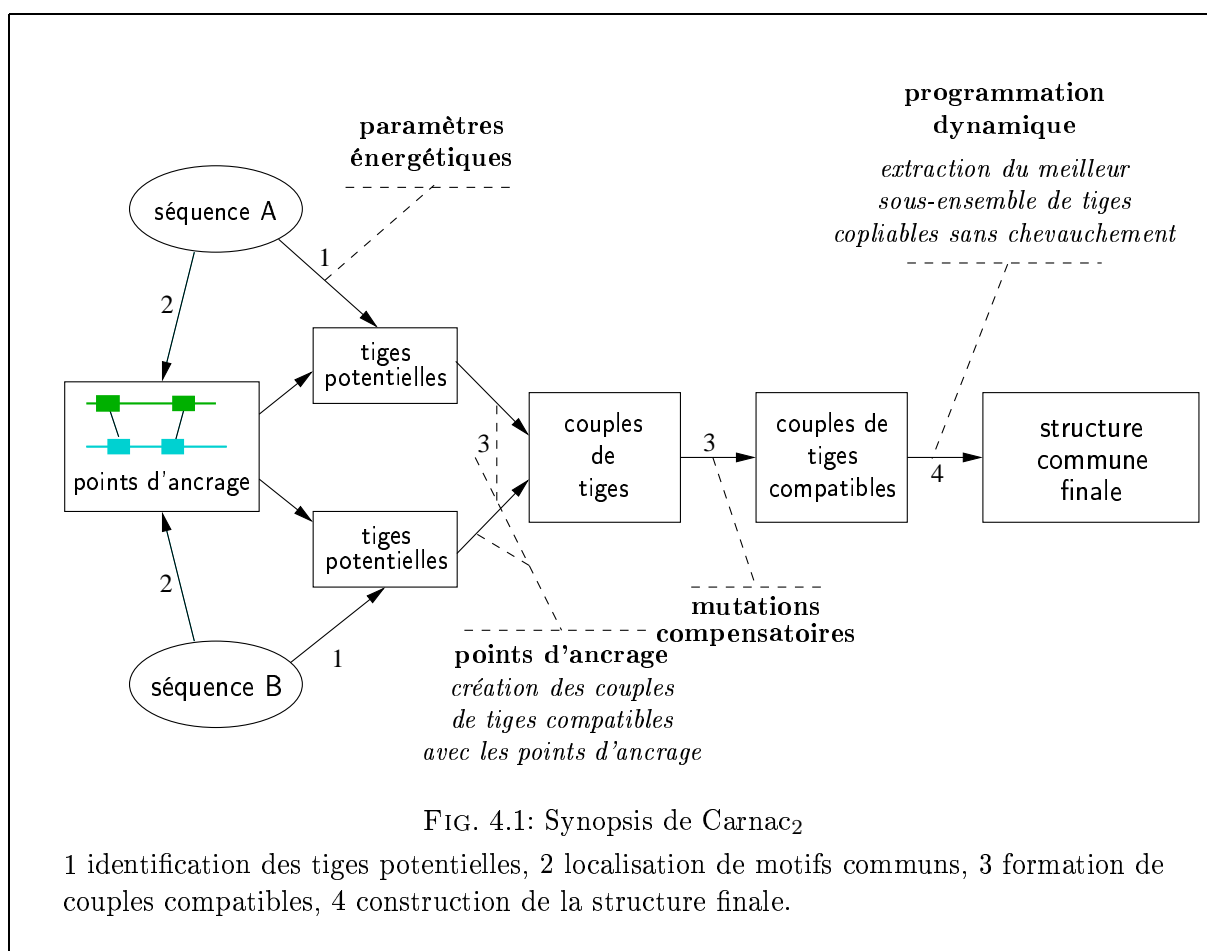
On retrouve bien l'organisation en feuille de trèfle. Tous les appariements prédits sont corrects, et la structure secondaire proposée ne diffère de la structure exacte que par un appariement absent. Mais ces méthodes de minimisation de l'énergie libre se heurtent à un problème inhérent à l'approche thermodynamique : l'existence de nombreuses structures alternatives avec des niveaux d'énergie très proches. Par exemple, pour la séquence de RNase P de *D. desulfuricans*, Mfold exhibe 18 structures potentielles globalement distinctes dont les niveaux d'énergie sont compris dans un intervalle de variation de 5%. De plus, la structure secondaire fonctionnelle *in vivo* peut être éloignée en terme d'énergie libre de l'optimalité. Cela s'explique par les contributions

énergétiques des interactions tertiaires, négligées par le modèle, mais également par l'influence de facteurs extérieurs qui échappent à l'approche thermodynamique, comme la cinétique du repliement ou le rôle de chaperons.

Les prédictions les plus fiables sont obtenues par *analyse comparative*. Dans le cas d'un ensemble de séquences homologues, la conservation de la structure au fil de l'évolution indique que la structure est fonctionnellement importante. L'analyse comparative s'intéresse à la prédiction de la structure dans ce contexte. Elle exploite le phénomène des mutations compensatoires, introduites en section 2.3.2. Quand on dispose d'un alignement de séquences homologues, partageant la même structure, il est possible d'identifier les couples de positions pour lesquelles les mutations se compensent. Cette covariation est mise en évidence en utilisant des mesures de corrélation, telles que le χ^2 ou l'information mutuelle. La construction de la structure secondaire ou tertiaire se fait ensuite par intégration des positions corrélées, suivant une stratégie gloutonne ou globale [18, 33, 31, 149]. Ce sont des approches de ce type qui ont permis de déterminer la structure des ARNt, ARNr 5S, ARNr 16S [89, 39, 150]. Les limites tiennent à des conditions d'application restrictives : il faut disposer d'un jeu de données de taille suffisante pour lequel la conservation de la structure secondaire est presque parfaite et la structure primaire a suffisamment peu divergé pour pouvoir construire un alignement multiple correct à la base près.

Plutôt que de s'opposer, l'approche thermodynamique et l'analyse comparative sont en fait parfaitement complémentaires. La première travaille sur les propriétés intrinsèques d'une séquence. La seconde tire parti des informations transverses et de la conservation de la structure entre les séquences. Il est donc naturel de chercher à les concilier afin de produire des prédictions de meilleure qualité qu'avec la seule minimisation de l'énergie libre dans un contexte d'utilisation plus souple que celui de l'analyse comparative classique. Cela a donné lieu au développement depuis quelques années d'un courant de méthodes hybrides.

On peut classer ces méthodes grossièrement en deux groupes suivant leur contexte d'application. Les méthodes du premier groupe utilisent toujours un alignement multiple sur la structure primaire comme point de départ : il s'agit par exemple de Pfold [79, 80], à base de grammaires stochastiques, de DC-Fold [129], qui applique une stratégie gloutonne de type diviser pour régner, d'X2S [72] ou de RNAalifold [58]. Les méthodes du second groupe s'affranchissent totalement de l'alignement préalable, pour pouvoir traiter des données médiocrement conservées qui ont trop divergé pour bien s'aligner. Il s'agit par exemple de Foldalign [47], RNAGA [16], Dynalign [97] et de Carnac [107, 135], dont nous allons parler en détail. Ce cadre d'application est le plus proche de la nature de l'ARN, où les niveaux de structure secondaire et tertiaire sont prépondérants par rapport à la structure primaire. Le prix à payer est la difficulté algorithmique. Le problème général d'inférer une structure secondaire commune à un ensemble de séquences à partir d'une liste d'appariements potentiels est NP-complet. Les différents programmes de prédiction proposent donc des solutions heuristiques. RNAGA est à base d'algorithmes génétiques. Foldalign et Dynalign sont des variations autour de l'algorithme de Sankoff, qui permet de construire une structure commune optimale en alignant et repliant les séquences simultanément [117]. Le coût de l'algorithme exact est prohibitif : il est exponentiel en fonction du nombre de séquences. Ces adaptations ne sont praticables qu'en imposant des limites sur la taille des données ou la forme des structures. Ainsi Foldalign est-il limité à la prédiction de structures sans multiboucles et Dynalign à un couple de séquences de quelques centaines de nucléotides. Carnac s'inscrit dans cette école de pensée, en dépassant l'écueil de la complexité. Cela est possible en cherchant à limiter l'espace de recherche à partir d'indices biologiques. De ce fait, Carnac est moins sensible que Dynalign, par exemple. Mais les résultats expérimentaux de la dernière section montrent sa robustesse et sa grande sélectivité.



4.2 Carnac

Comme la plupart des méthodes de prédiction, Carnac utilise trois sources d'information : l'énergie libre de la structure, les mutations compensatoires et la conservation éventuelle au niveau de la séquence. L'originalité vient de la stratégie de prédiction mise en œuvre pour sélectionner les tiges. Nous proposons un algorithme en deux étapes : prédiction deux à deux sur un schéma proche de Sankoff, puis combinaison des prédictions. Dans cette construction, les optimisations tirent parti du caractère biologique des données. La philosophie est de privilégier la sélectivité des prédictions, avec un faible nombre de faux positifs.

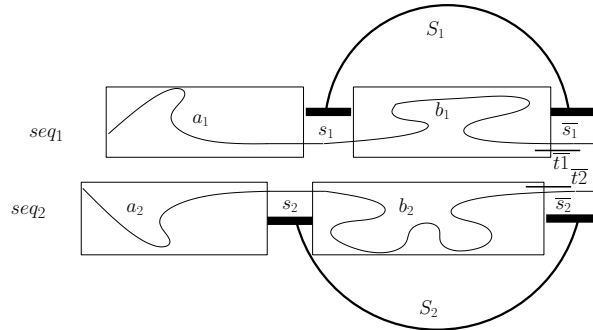
4.2.1 Prédiction deux à deux

Carnac₂ est la première étape de Carnac. C'est une heuristique qui permet d'inférer une structure commune à deux séquences. La figure 4.1 résume le déroulement.

La structure commune est construite en se plaçant au niveau des tiges et non des bases, pour des raisons d'efficacité. Le premier travail est donc d'annoter chaque séquence en identifiant les tiges potentielles maximales (étape 1 sur la figure 4.1). Cette recherche utilise les paramètres énergétiques du modèle thermodynamique de Turner et al. [65]. Le seuil de sélection des tiges est une fonction affine par morceaux, qui dépend de la distance entre l'ouverture et la fermeture de la tige, et prend en compte le biais de contenu en GC.

À partir de ces tiges brutes, on limite l'espace de recherche pour les structures communes potentielles en constituant un sous-ensemble de *couples de tiges compatibles*. Ce sont les phases 2 et 3 de la figure 4.1. La compatibilité d'une tige de la séquence 1 avec une tige de la séquence 2 dépend de propriétés des séquences et des tiges. Carnac₂ s'appuie sur les éventuelles conservations au niveau de la structure primaire. L'algorithme repère les zones exceptionnellement bien conservées au niveau de la séquence. Ces régions servent de *points d'ancrage* pour guider le repliement commun (la philosophie est la même qu'en section 3.6). Une fois obtenus les points d'ancrage, les deux jeux de tiges sont filtrés pour former des couples de tiges candidats. Il y a deux contraintes pour que des tiges soient compatibles. La première est de respecter les informations de la structure primaire : l'alignement induit par les tiges ne doit pas contredire l'alignement local des séquences défini par les points d'ancrage. La seconde est de montrer des indices d'évolution commune : les tiges doivent présenter au moins une mutation compensatoire. Cela a, de plus, pour effet bénéfique de favoriser la formation de tiges hors des régions conservées, ce qui est cohérent avec les matrices RIBOSUM [78].

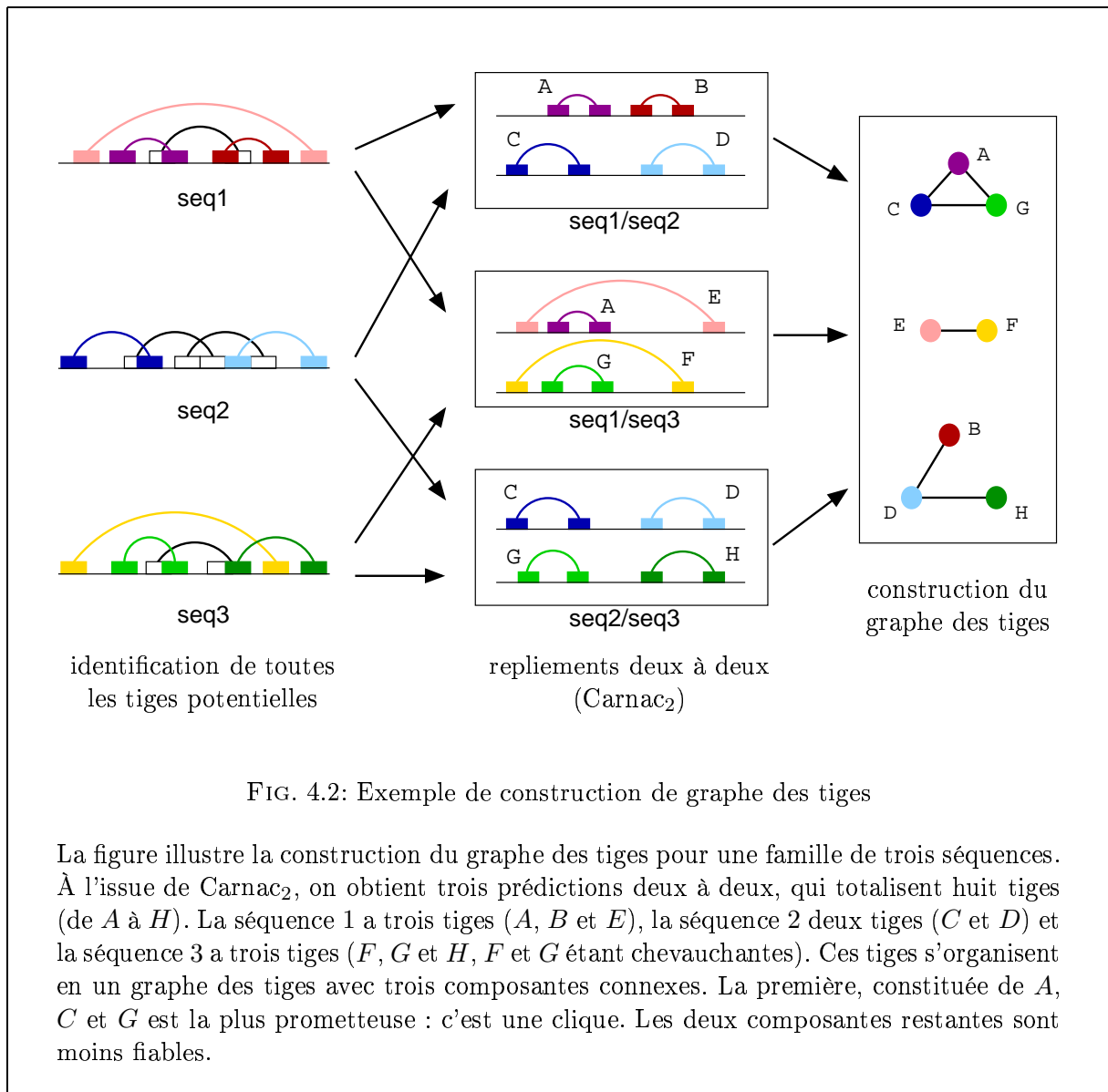
Une fois cet ensemble de couples de tiges candidats construits, il reste à en extraire le sous-ensemble d'énergie optimale constituant une structure secondaire. L'opération emprunte le schéma de l'algorithme original de Sankoff [117]. L'algorithme de Sankoff construit une structure commune sous forme d'arbre d'alignement, où les nœuds sont une base ou une paire de bases. Cette approche exacte à une complexité $O(n^6)$ en temps et $O(n^4)$ en espace, ce qui la rend non opérationnelle. Nous reprenons ce principe avec l'heuristique qui consiste non pas à former des paires de bases, mais des couples de tiges, et à limiter les insertions et délétions de tiges aux tiges-boucles. En d'autres termes, Carnac₂ travaille sur un arbre compacté, où les nœuds sont des tiges et les insertions et les délétions portent sur nœuds terminaux uniquement.



$$\alpha(seq_1, seq_2) = \min \begin{cases} \alpha(a_1, a_2) + \alpha(b_1, b_2) + \text{énergie}(S_1, S_2) & (1) \\ \alpha(a_1, a_2 s_2 b_2 \bar{s}_2) + \text{énergie}(S_1, -) & (2) \\ \alpha(a_1 s_1 b_1 \bar{s}_1, a_2) + \text{énergie}(-, S_2) & (3) \\ \alpha(a_1 s_1 b_1 \bar{s}_1, a'_2 t_2 b'_2 \bar{t}_2) & (4) \\ \alpha(a'_1 t_1 b'_1 \bar{t}_1, a_2 s_2 b_2 \bar{s}_2) & (5) \end{cases}$$

α désigne l'énergie du repliement commun, c'est-à-dire la somme des énergies des tiges constituant ce repliement, données par la fonction *énergie*.

Dans le cas (1), les deux tiges S_1 et S_2 sont sélectionnées et alignées : on crée le nœud (S_1, S_2) dans l'alignement. Dans le cas (2), S_1 est sélectionnée comme tige célibataire, sans contrepartie dans la séquence 2 : on crée le nœud $(S_1, -)$. Cette possibilité n'est offerte que si S_1 est une tige-boucle, ce qui est géré par la fonction *énergie*. Le cas (3) est symétrique de (2) : S_2 est une tige-boucle sélectionnée comme tige célibataire, ce qui crée le nœud $(-, S_2)$. Dans les deux dernier cas (4) et (5), une des deux tiges S_1 ou S_2 , au moins, n'est pas sélectionnée dans la



structure commune. (t_1, \bar{t}_1) et (t_2, \bar{t}_2) désignent les prochaines tiges potentielles qu'on rencontre dans l'ordre lexicographique sur les positions de fermeture et d'ouverture.

Dans ce calcul, seuls les couples de tiges compatibles sont considérés, ce qui réduit le volume des données à traiter. On optimise ainsi l'espace mémoire nécessaire, en travaillant autour de l'hyperdiagonale. La mise en œuvre se fait donc par programmation dynamique avec une structure de graphe en guise de table.

4.2.2 Combinaison des repliements

Comment passer de 2 à un nombre quelconque, n , de séquences? À l'issue de tous les repliements deux à deux, on obtient $n(n-1)/2$ structures distinctes, soit $n-1$ prédictions par séquence. Il faut concilier ces différentes prédictions, en dégagant un critère de sélection pour différencier les faux positifs des tiges correctement prédites.

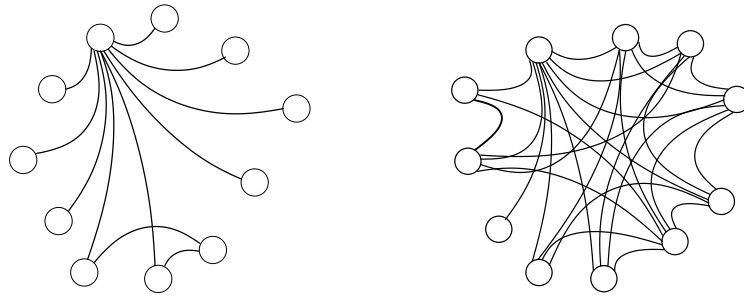


FIG. 4.3: Graphe d'une tige attractive (à gauche), comparé à un graphe fortement connecté (à droite)

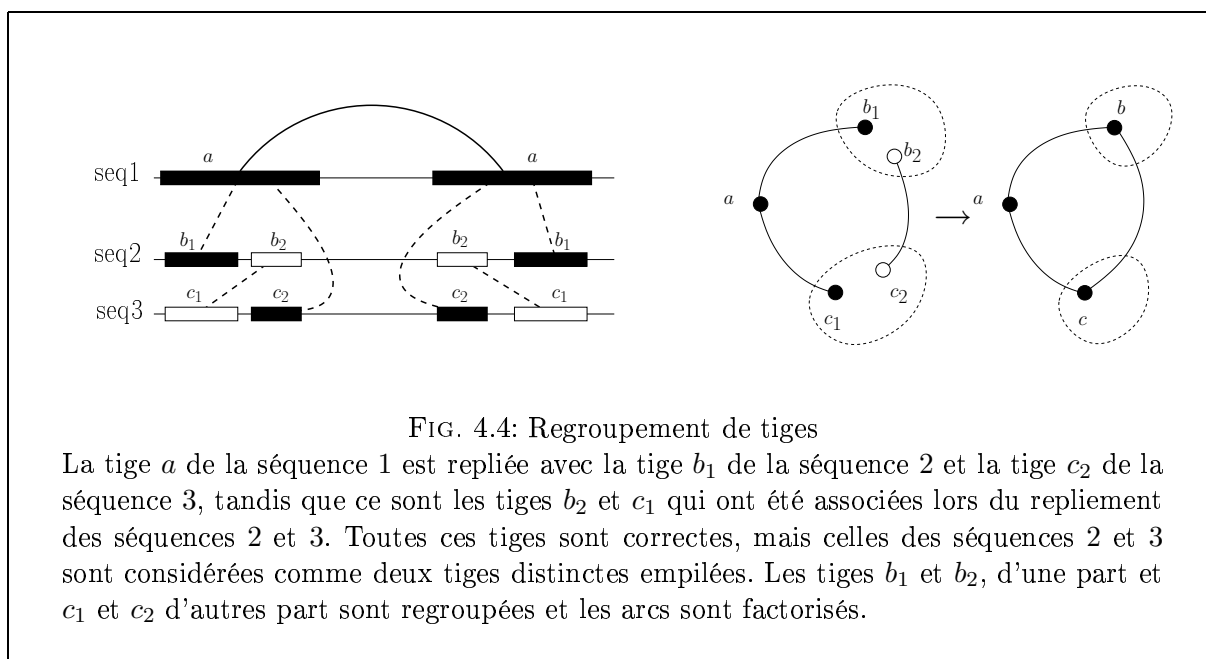
Un premier indice est le nombre de fois où une tige est retenue au fil des différentes prédictions de Carnac₂. Mais plusieurs tests expérimentaux ont montré que la seule fréquence d'apparition n'était pas une information suffisante. Il faut se prémunir du phénomène des tiges attractives : une tige A avec un très bon niveau d'énergie est de ce fait sélectionnée à chaque repliement, avec une tige B_i de la séquence i . Pour que ces repliements soient cohérents, il faut que les tiges B_i soient également associées entre elles. C'est-à-dire que si une tige A de la séquence 1 est repliée avec une tige B de la séquence 2 d'une part, et avec une tige C de la séquence 3 d'autre part, on s'attend à ce que la relation soit transitive et que B et C forment un couples de tiges dans la structure commune des séquences 2 et 3. Cela conduit à prendre en compte la topologie de l'ensemble des prédictions. Cette idée est mise en œuvre avec le *graphe des tiges*. C'est un graphe non orienté, où chaque tige est un nœud et il existe un arc entre une tige A de la séquence i et une tige B de la séquence j si le couple (A, B) est sélectionné dans la structure commune des séquences i et j produite par Carnac₂. La figure 4.2 illustre la construction du graphe des tiges pour un jeu de trois séquences. Les tiges prédites avec le plus de régularité apparaissent dans des composantes connexes fortement connectées. Les tiges attractives sont dans des composantes «en étoile» (figure 4.3).

Ce graphe des tiges brut subit ensuite deux modifications, destinées à prendre en compte les empilements de tiges d'une part, et à corriger les effets de l'absence de covariations d'autre part. On regroupe les petites tiges bruitées empilées (figure 4.4). Puis le graphe est enrichi pour introduire des informations concernant les covariations, ou plutôt l'absence de covariations. Aux arcs existants, de type dit *coplié*, sont ajoutés des arcs, de type *identité*, entre les tiges ne présentant aucune covariation. Un arc *identité* n'est pas informatif en terme de repliement commun, mais il ne doit pas non plus pénaliser la connectivité du graphe.

La composante idéale est une clique comportant autant de nœuds que de séquences, les différents nœuds correspondant à des classes de tiges dans des séquences différentes. Nous définissons deux indices : *node_index*, qui mesure l'écart par rapport à ce cas idéal au niveau des nœuds, et *edge_index* au niveau des arcs.

$$node_index = \left(\frac{N_s - (N - N_s)}{n} \right)^2 \quad edge_index = \frac{co}{me - id}$$

où N est le nombre de nœuds de la composante connexe, N_s le nombre de séquences différentes



présentes dans la composante, n le nombre total de séquences, co le nombre d'arcs de type *coplié*, id le nombre d'arcs de type *identité*, et $me = N(N - 1)/2$ le nombre maximum d'arcs possibles.

Pour la formation de la structure finale, chaque tige se voit attribuer l'indice de sa composante. Puis pour chaque séquence les tiges sont incorporées de manière glotonne par indice décroissant jusqu'à un seuil donné. Nous avons essayé une approche plus fine, de type Nussinov et Jacobson [103] avec les indices comme poids, sans bénéfice. Cela montre qu'à ce stade de l'algorithme la structure commune est déterminée sans ambiguïté.

4.3 Résultats expérimentaux

Carnac est implémenté en C et dispose d'un site web ² qui permet une utilisation souple. La visualisation des résultats utilise Naview [12] ou le programme maison RNafamily dédié à la représentation de structures homologues.

La fonctionnalité de Carnac a été établie sur plusieurs jeux de données. Nous commençons par reprendre une étude comparative des méthodes de prédiction menée dans [42], qui inclut Carnac. C'est l'objet des figures 4.5 et 4.6. Ces données de benchmark concernent huit familles de séquences présentant un taux de similitude élevé (de 80% à 90%) ou moyen (de 60% à 80%). Elles montrent que Carnac est particulièrement efficace et sélectif.

Nous complétons cet batterie de tests avec trois exemples sur des séquences avec un taux de similitude plus bas, qui ont fortement divergé d'une espèce à l'autre. C'est le domaine d'application typique de Carnac, où les méthodes de prédiction nécessitant un alignement préalable ne peuvent plus s'appliquer. Il s'agit d'une famille de RNase P qui reprend l'exemple de la section 2.3 (figure 4.7), d'une famille de télomérases de ciliés (figure 4.8) et d'une famille d'ARN ribosomiques (figure 4.9). Le dernier jeu de données ouvre de nouvelles perspectives. Il s'agit de séquences partiellement structurées (des parties 5'UTR d'enterovirus). Dans ce contexte, Carnac

²<http://bioinfo.lifl.fr/carnac>

jeu de données	longueur	% identité deux à deux		nombre de séquences	
		élevé	moyen	élevé	moyen
LSU rRNA (<i>E. coli</i>)	2904	88.1	72.0	11	11
SSU rRNA (<i>E. coli</i>)	1542	90.7	80.0	11	11
RNase P (<i>E. coli</i>)	377	81.5	67.1	9	11
tRNA-phe (<i>S. cerevisiae</i>)	73	84.4	60	11	11

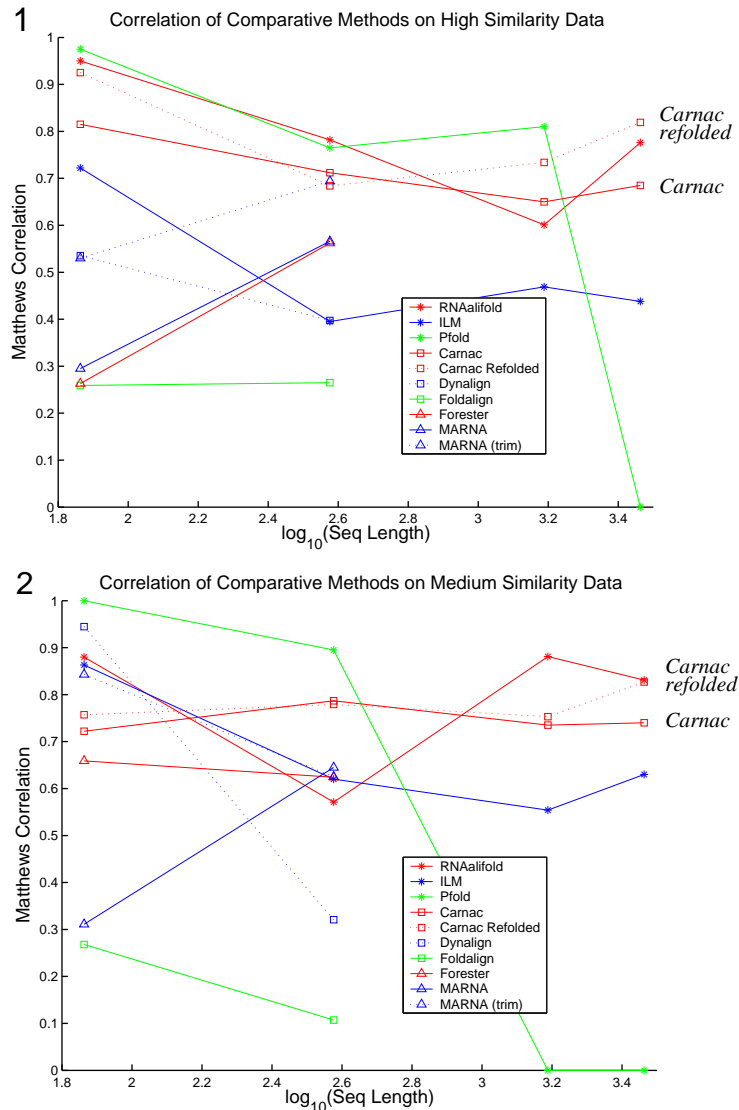


FIG. 4.5: Comparaison des méthodes d'analyse comparative I (d'après [42])

L'ensemble de test est constitué de huit jeux de données pour lesquels les structures secondaires et tertiaires sont bien établies. Quatre ont un taux de similitude élevé et quatre un taux de similitude moyen, avec une longueur variant de 70 à près de 3000 bases (tableau du haut). Les résultats obtenus avec chacun des programmes sur chacun des jeux de données sont évalués par le coefficient de corrélation de Matthews [5], qui prend en compte la sensibilité et la sélectivité des prédictions. Il apparaît que la longueur des séquences est un facteur limitant pour la plupart des méthodes. Ces diagrammes montrent clairement le bon comportement de Carnac. Il fait jeu égal avec Alifold, qui travaille à partir d'un alignement multiple. *Carnac refolded* correspond à une option de Carnac où la structure est prédite en deux étapes : un premier passage avec Carnac produit une première structure où toutes les tiges sélectionnées doivent montrer des covariations, puis cette structure est complétée par un second passage en relaxant la condition sur les mutations compensatoires.

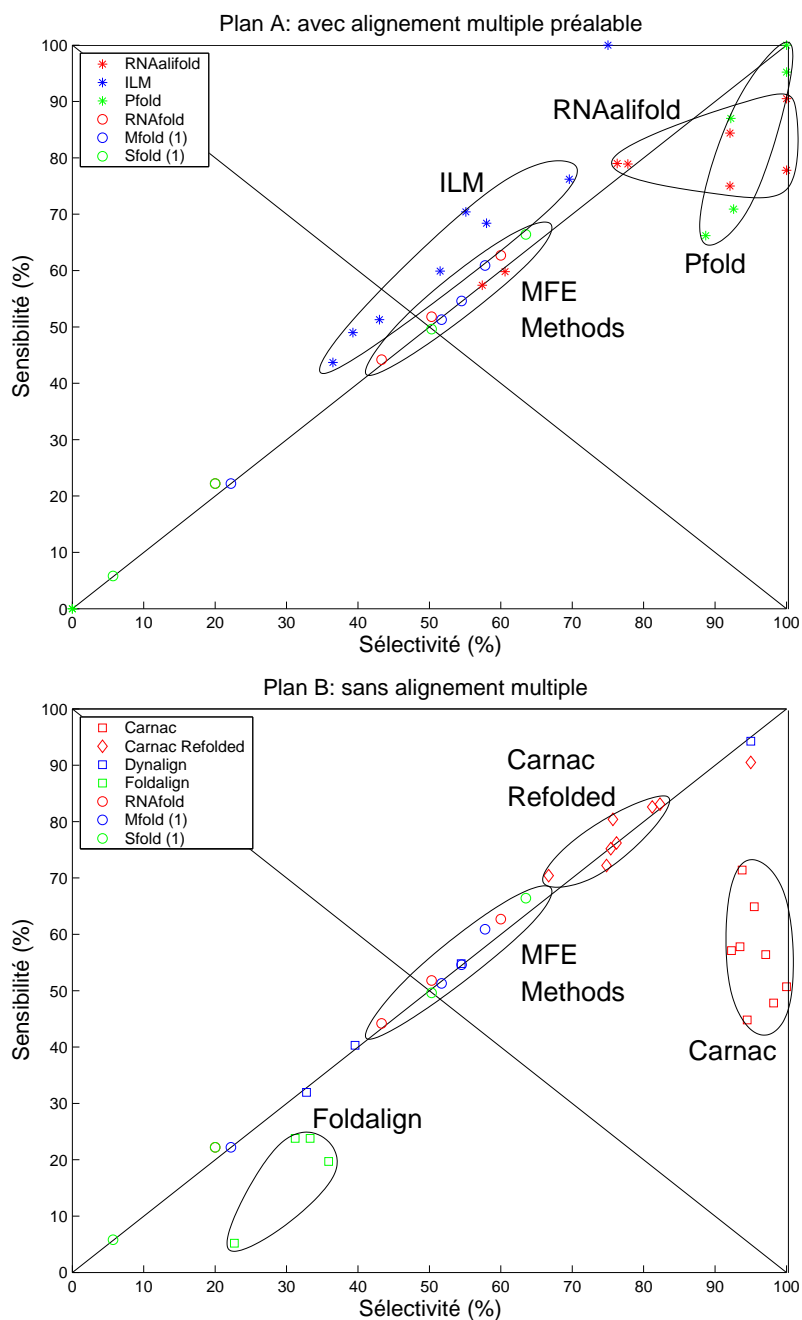


FIG. 4.6: Comparaison des méthodes d'analyse comparative II (d'après [42])

Cette figure complète la figure 4.5. Elle distingue les programmes qui travaillent à partir d'un alignement multiple préalable supposé correct (*plan A* : RNAalifold, Pfold) des programmes qui ne demandent pas d'alignement multiple (*plan B* : Foldalign, Dynalign et Carnac). Sur les deux graphiques, les méthodes classiques de minimisation de l'énergie libre (Mfold, RNAfold et Sfold) sont indiquées comme référence. Cela fait apparaître que Carnac est la méthode de choix pour le plan B. De plus, sa sélectivité est comparable à celle des algorithmes du plan A.

Carnac

nom	appariements	vrais positifs	faux négatifs
<i>D. desulfuricans</i>	84	79 (94%)	27%
<i>D. vulgaris</i>	74	69 (93%)	36%
<i>G. sulfurreducens</i>	80	68 (85%)	39%
<i>C. jejuni</i>	61	54 (88%)	41%
<i>H. pylori</i>	57	51 (89%)	40%

RNAga

nom	rang	appariements	vrais positifs	faux négatifs
<i>D. desulfuricans</i>	1	90	64 (71%)	41%
<i>D. vulgaris</i>	2	91	53 (58%)	51%
<i>G. sulfurreducens</i>	4	102	67 (66%)	39%
<i>C. jejuni</i>	10	79	43 (55%)	52%
<i>H. pylori</i>	1	84	40 (48%)	52%

Foldalign

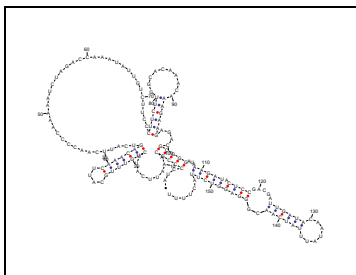
nom	appariements	vrais positifs	faux négatifs
<i>D. desulfuricans</i>	12	5 (41%)	95%
<i>D. vulgaris</i>	12	5 (41%)	95%
<i>G. sulfurreducens</i>	12	5 (41%)	95%
<i>C. jejuni</i>	13	5 (38%)	94%
<i>H. pylori</i>	10	5 (50%)	94%

FIG. 4.7: Résultats de Carnac pour une famille de RNase P

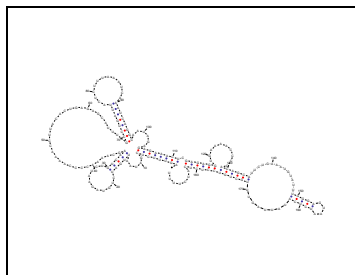
Ce jeu de données est composé de cinq séquences de RNase P de la famille Delta/Epsilon Purple Bacteria [11]. Il comprend notamment les deux séquences de la figure 2.5 qui ne s'alignent pas. Les séquences font 350 bases de long. Elles présentent une structure consensus composée de 16 à 17 tiges, qui varie selon les espèces (voir figure 2.5). La conservation au niveau de la structure primaire est plus faible que pour l'exemple de la figure 4.5 (environ 60%) et la présence de deux pseudo-nœuds fait que les méthodes thermodynamiques classiques, [158] ou [57], s'appliquent mal. Nous comparons les résultats de Carnac à deux autres méthodes qui travaillent dans le même contexte, sur des séquences non-alignées : RNAga [17] et Foldalign [47]. Pour chaque repliement sont indiqués le nombre d'appariements prédits, le nombre et le pourcentage de vrais positifs, et le pourcentage de faux négatifs. Pour RNAga, nous avons sélectionné la meilleure structure parmi les dix proposées en indiquant son rang.

1. Prédictions de Carnac

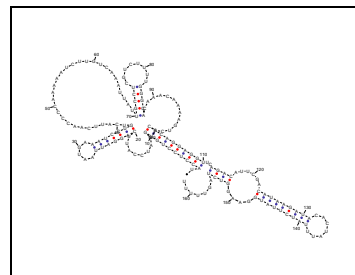
AF417611/283-441



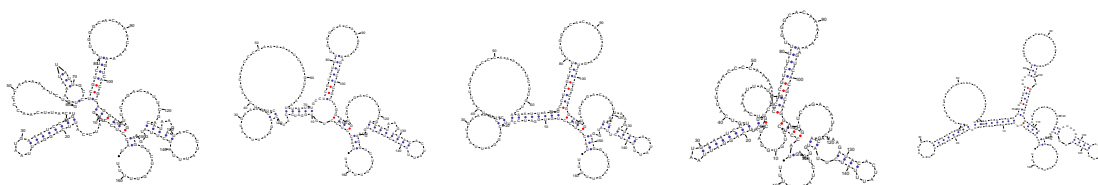
U10565/50-238



AF417612/231-392



2. Prédictions de X2S AF417612/231-392 (itérations 1 à 5)



3. Résultats comparés de Carnac, RNAGA et Mfold sur la séquence AF417612/231-392

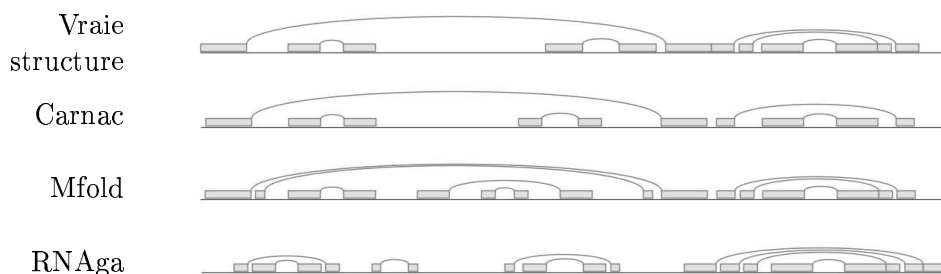
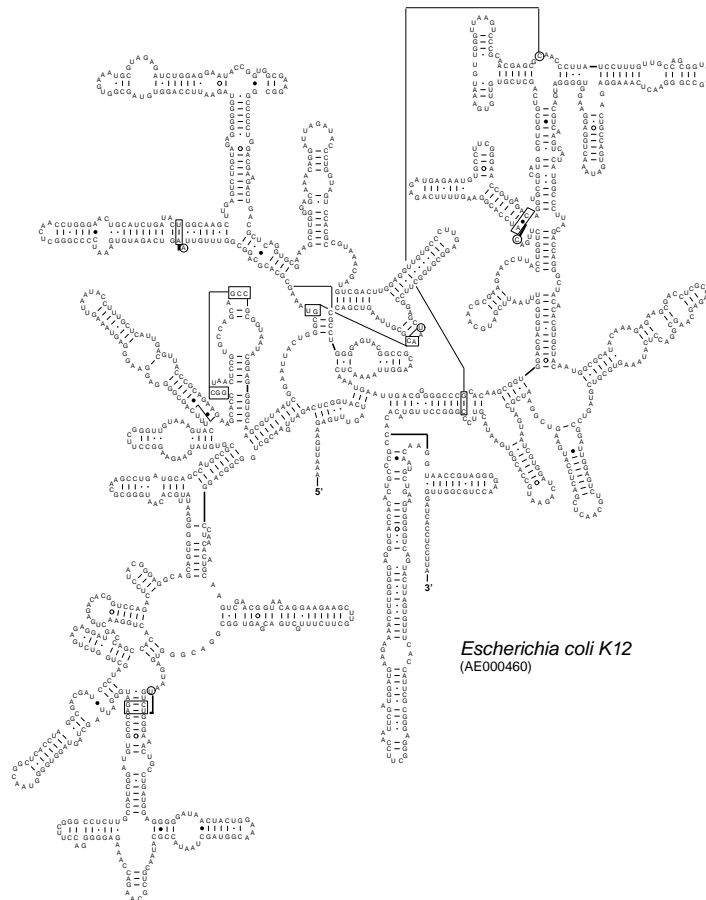


FIG. 4.8: Résultats de Carnac pour des télomérases de ciliés

Le télomérase est un complexe ribonucléoprotéique qui permet l'élongation et la conservation de la taille des télomères. Il est constitué d'un assemblage entre une transcriptase inverse renfermant une sous-unité catalytique et une molécule d'ARN. Les ARN du télomérase diffèrent beaucoup en séquence et en structure entre les différents domaines. La banque RFAM propose 17 séquences, dont le taux d'identité est égal à 58%. Lorsqu'on tente de les aligner par des algorithmes classiques d'alignement multiple (avec Clustal ou Dialign2), ces séquences s'alignent très mal. Les prédictions proposées par PFOLD, RNAalifold (non montrés) et X2S qui sont tributaires de la qualité de l'alignement multiple initial, sont de ce fait totalement erronées. Nous avons appliqué Carnac sur trois séquences, qui semblent présenter le plus de disparités (AF417611/283-441, U10565/50-238, AF417612/231-392). Il apparaît que RNAGA ne trouve pas la structure. Les prédictions de Carnac et Mfold sont en accord avec la structure consensus disponible dans [49].



organisme	Mfold					Carnac		
	sous-opt.	rang	app. pred.	vrais pos.	faux neg.	app. pred.	vrais pos.	faux neg.
<i>M.jannaschii</i>	14	9	494	294 (59%)	35%	325	257 (79%)	44%
<i>S.solfataricus</i>	16	7	495	287 (57%)	38%	355	300 (84%)	35%
<i>E.coli</i>	27	21	476	340 (71%)	28%	321	278 (86%)	41%
<i>B.subtilis</i>	17	5	489	271 (55%)	42%	324	268 (82%)	43%

FIG. 4.9: Résultats de Carnac pour une famille d'ARNr 16S

Les ARN ribosomiques 16S sont des constituants essentiels de la petite sous-unité du ribosome présents chez toutes les bactéries. Ils partagent une structure consensus commune, qui varie toutefois au fil des espèces en fonction de la distance évolutive. La séquence fait approximativement 1500 nucléotides, et la structure consensus comporte environ 80 tiges. Nous avons choisi une espèce représentative de chaque domaine de façon à tester la robustesse de Carnac face à ces variations [151]. Pour Mfold, figurent le nombre de structures sous-optimales proposées, et pour la meilleure d'entre elles, son rang parmi les sous-optimales ainsi que le nombre d'appariements prédits et les pourcentages de vrais positifs et faux négatifs. Il apparaît que Carnac est particulièrement sélectif, tout en ayant un taux de couverture plus faible que Mfold.

permet de détecter la présence ou l'absence d'une structure commune, si on fait l'hypothèse que les séquences sont homologues (figure 4.10).

Ces exemples démontrent la robustesse de Carnac face à des données avec un faible niveau de conservation. Carnac prend le relais des méthodes comparatives quand l'obtention d'un alignement multiple est problématique. Dans ce cadre, les prédictions obtenues sont globalement aussi sélectives, au détriment toutefois de la sensibilité.

Enfin, les optimisations algorithmiques mises en œuvre s'avèrent suffisamment efficaces pour pouvoir traiter dans de bonnes conditions de longues séquences, ce qui était une des motivations initiales. Le temps de calcul de Carnac est fonction de la taille du jeu de données et de la distance évolutive entre les séquences. Pour donner une estimation, les différents jeux de données présentés ici nécessitent en temps CPU moins de 1 seconde pour le jeu de télomérases, 15 secondes pour les ARN RNase P et quelques minutes pour les 12 séquences d'entérovirus de 1800 bases de long (sur un PC de bureau).

4.4 Perspectives

Les travaux présentés dans ce chapitre traitent de la prédiction de la structure secondaire des ARN dans le cadre classique de l'analyse comparative, en présence d'une famille de gènes à ARN homologues. Mais les nouvelles données apportées par les programmes de séquençage apportent un éclairage qui renouvelle la problématique de l'inférence de structure.

Une première direction de recherche est motivée par l'étude des ARN messagers. Différents travaux mettent en évidence la formation de courts motifs structuraux qui participent au contrôle transcriptionnel et post-transcriptionnel de l'expression des gènes. Ce phénomène est connu chez les bactéries, avec la formation de tiges boucles comme terminateur de la transcription dans la partie 3' non traduite de l'ARNm. Des motifs peuvent apparaître dans la région 5' non traduite, comme les éléments IRE (iron response element) qui participent à la stabilité du messager via la liaison à des ions [98]. Ils peuvent également se former au sein des ORF et dans ce cas ils influent sur le processus de traduction, en induisant un décalage du cadre de lecture par exemple [7]. Les méthodes de prédiction existantes ne sont pas armées pour analyser des séquences qui présentent des éléments structurels locaux. Il y a donc un besoin d'algorithmes de recherche et de découverte de signaux liés par une structure commune sans conservation de séquence. La difficulté est accentuée par le fait que ces motifs contiennent souvent des pseudo-nœuds. Le logiciel ComRNA [67] est un premier outil qui permet la gestion de pseudo-nœuds sans alignement préalable, mais il ne s'applique qu'à de courtes séquences suffisamment conservées.

Un second problème est de décider si une séquence présente effectivement une structure fonctionnelle. La comparaison à grande échelle entre les génomes humain, de rat et de souris fait apparaître plusieurs dizaines de clusters de séquences conservées qui ne sont pas annotées comme codantes et qui sont des candidats pour être des gènes à ARN [6]. Comment discriminer entre les gènes à ARN et les gènes codants? Dans le cas d'une seule séquence, cette question reste ouverte. Le modèle thermodynamique ne permet pas de distinguer entre le codant et le non-codant, faute de signal statistiquement valide [152]. Dans le cas particulier d'alignements homme/souris, le programme QRNA propose une approche par apprentissage à partir de l'analyse de la distribution des mutations [110] : les substitutions observées dans l'alignement sont-elles des mutations silencieuses au regard du code génétique ou peuvent-elles plutôt s'interpréter comme des mutations compensatoires? Ce travail est pionnier dans le sens où c'est le premier modèle proposé, mais les résultats restent perfectibles. Comme le montre la figure 4.10, Carnac

permet partiellement d'aborder cette question dès lors que l'on dispose d'un nombre suffisant de séquences, mais deux séquences ne suffisent pas.

Cette nouvelle thématique de prédiction/détection de la structure pose comme problème corollaire l'évaluation de la significativité des observations. C'est un aspect particulièrement sensible quand on prédit un motif structural local de quelques dizaines de nucléotides seulement, ou une structure globale commune pour des séquences avec un niveau de conservation très élevé. Un tel projet implique d'avoir des modèles d'évolution des ARN et des modèles de génération aléatoire. On rejoint là les perspectives du chapitre précédent sur la comparaison de structures.

Chapitre 5

Analyse des régions régulatrices

Après l'étude des structures d'ARN, ce chapitre aborde un second thème bio-informatique : l'analyse des signaux de régulation qui participent au niveau de l'ADN au contrôle de la transcription. Ces travaux ne portent pas sur la modélisation ou la découverte des motifs associés, mais plutôt sur leur localisation et leur combinaison. Si on souhaite raisonner à l'échelle d'un génome, cela pose des problèmes algorithmiques spécifiques à cause de la taille des données et des biais de composition. Ce sont des recherches en cours, qui ont été initiées avec la thèse de Matthieu Defrance débutée en 2003 et la thèse d'Aude Liefoghe démarrée cette année, en 2004. Elles s'inscrivent dans un projet général pour l'analyse des régions régulatrices dans les génomes d'eucaryotes supérieurs (avec la suite TFM-Scan, TFM-Explorer et TFM-Database ³).

5.1 La régulation transcriptionnelle

La régulation de l'expression des gènes permet à la cellule de se différencier et d'avoir une réponse adaptée à son environnement. L'activité d'un gène et la synthèse d'une protéine sont contrôlées à plusieurs niveaux : l'initiation de la transcription, qui conduit à la synthèse de l'ARNm, l'épissage, le transport d'ARNm, l'initiation de la traduction, les modifications post-traductionnelles, la dégradation des ARNm et des protéines. Dans cette multiplicité de mécanismes, nous mettons l'accent sur l'étape préalable d'initiation de la transcription.

La transcription est déclenchée par la présence de l'ARN polymérase et implique l'intervention de protéines auxiliaires spécifiques, les facteurs de transcription. Les facteurs de transcription agissent en se fixant à l'ADN en impliquant un domaine de liaison qui présente une affinité pour un court motif nucléique, un *site de fixation*, dans la région promotrice. La nature et le nombre de facteurs de transcription fixés sur un promoteur déterminent la spécificité et l'intensité de transcription [106].

L'initiation de la transcription demande ainsi l'assemblage d'un complexe multiprotéique, ce qui ouvre la porte à des mécanismes régulateurs sophistiqués. Les sites de fixation peuvent se trouver à de multiples endroits par rapport au promoteur du gène. Chez les eucaryotes, on a mis en évidence l'existence d'éléments situés plusieurs milliers de bases en amont du site d'initiation de la transcription ou dans les introns, souvent en synergie avec des éléments plus proches. Au niveau génomique, en dimension un, la régulation transcriptionnelle se traduit donc

³TFM : TF signifie *Transcription Factor*, et le M est pour *Module* ou *Matrix* ou *Magic* ou *Matthieu*, car c'est Matthieu Defrance qui le premier s'est attelé à ce travail, avec TFM-Explorer

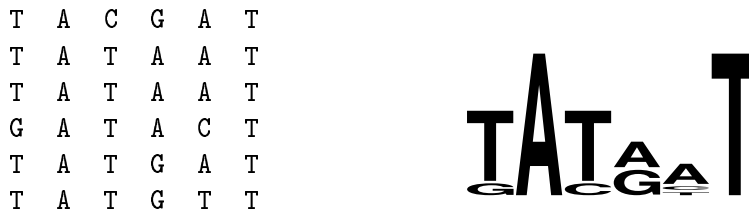


FIG. 5.1: Séquence consensus pour la boîte de Pribnow (à partir de [127])

La boîte de Pribnow est un signal d'environ 10 nucléotides situé en amont du site d'initiation de la transcription qui interagit avec l'ARN polymérase chez les procaryotes. La figure montre un échantillon de 6 sites. La première séquence consensus possible pour cet alignement est TATAAT, formée en relevant la base la plus fréquente à chaque position. Cela ne capture qu'un tiers des sites. Si on autorise une substitution, on attrape la moitié des sites, avec une occurrence toutes les 200 bases en moyenne. Pour reconnaître les 6 sites, il faut autoriser deux substitutions et dans ce cas, le motif apparaît toutes les 30 bases en moyenne. Une solution alternative est de partir d'un consensus plus souple, TATRNT, qui permet d'identifier 4 des 6 sites. La fréquence associée est 500 bases. Avec une substitution, les 6 sites sont représentés, mais on obtient une occurrence toutes les 30 bases.

par une combinaison de signaux dans l'ADN, les sites de fixation. Cette information ne capture pas toute la complexité du phénomène, loin s'en faut. Elle ignore la structure de la chromatine, la méthylation de l'ADN, la dynamique de la fixation [50]. C'est toutefois un matériau qui se prête à un traitement bio-informatique et qui permet un premier pas vers la compréhension des mécanismes spécifiques qui conditionnent l'expression d'un gène.

5.2 Le modèle des matrices

Les sites de fixation pour un facteur donné ne sont pas des motifs exacts. Ils présentent des variations, qui participeraient à la modulation du contrôle de l'expression. Suivant l'affinité du site, et donc la qualité de l'interaction ADN-protéine, un même facteur peut ainsi jouer sur l'intensité de la transcription en fonction du gène.

Il faut un mode de représentation pour l'ensemble des sites de fixation associés à un même facteur. Les articles de revue [127, 128, 143] font le point sur cette question. Une séquence consensus simple, écrite à partir du code IUPAC, ne convient pas. L'exemple de la figure 5.1 montre les limites de cette approche. Les matrices sont des modèles probabilistes plus riches qui permettent d'obtenir un compromis entre la sensibilité et la sélectivité. Le point de départ de la construction d'une matrice pour un facteur de transcription est un échantillon de sites de fixation connus, regroupés dans un alignement multiple. Le score d'un mot s'interprète comme

$$\log_2 \left(\frac{\text{probabilité dans le modèle cible (alignement multiple)}}{\text{probabilité dans le modèle de fond (génomique)}} \right)$$

Deux hypothèses simplifient ce travail : la première est que les contributions énergétiques sont additives. Cela permet de représenter la spécificité par une matrice mono-nucléotide, sans prendre en compte les voisins. La deuxième est que le modèle de fond, le génome, peut être approximé par une séquence aléatoire générée par un modèle de Bernoulli. On a ainsi un système de score additif. On définit $w(b, i)$ le score de la base b à la position i .

$$w(b, i) = \log_2 \left(\frac{f_{b,i}}{p_b} \right) \quad (5.1)$$

où $f_{b,i}$ est la fréquence de la base b à la position i de l'alignement et p_b est la fréquence moyenne de la base b dans le génome. Les poids positifs bonifient les bases qui apparaissent plus que la moyenne dans le modèle, et les poids négatifs pénalisent les bases qui apparaissent moins que la moyenne. Les substitutions sont pénalisées en fonction de la conservation de la position. Cette définition des poids pour la matrice trouve également une justification thermodynamique : c'est la valeur qui maximise la probabilité de liaison à toutes les séquences de l'échantillon [8]. Le score d'un mot $u_1 \cdots u_n$ est calculé simplement :

$$\text{Score}(u_1 \cdots u_n) = \sum_{i=1}^n w(u_i, i).$$

Il peut être assorti du calcul de la P-valeur [125, 63], qui informe sur sa significativité. La P-valeur est ici la probabilité d'observer un score supérieur ou égal à une position donnée. On peut également rapprocher ce système de score de l'entropie relative (ou distance de Kullback-Leibler, ou contenu en information) de l'alignement multiple à la position i :

$$I_{seq}(i) = \sum_b f_{b,i} \log_2 \left(\frac{f_{b,i}}{p_b} \right). \quad (5.2)$$

L'entropie relative est égale au score moyen à la position i pour les séquences de l'échantillon d'apprentissage.

La pertinence de la représentation par matrice et du calcul de score associé a été établie *in vitro* pour différentes familles de facteurs de transcription. La très grande majorité des sites prédits se lie effectivement [136], et le score reflète la spécificité du site [114]. Il existe plusieurs bases de données de sites de fixation, assortis des matrices correspondantes, comme Transfac [148], Jaspar [116] ou Plantcare [88] pour les eucaryotes. La diffusion de ces données assure la popularité du modèle.

5.3 Localisation à grande échelle des sites de fixation

La disponibilité de nombreux génomes, et notamment de génomes eucaryotes, ouvre la voie à la recherche systématique de sites de fixation à grande échelle. On accède ainsi aux séquences des introns, aux positions distales dans les promoteurs. On peut extraire des gènes cibles potentiels à l'échelle d'un génome. Quels sont les gènes qui contiennent un site de fixation pour tel facteur à telles positions ? Quels sont les gènes qui présentent une association de facteurs donnés dans leur voisinage ?

Les algorithmes mis en œuvre dans les programmes usuels, tels que Patser [157] ou Matinspector [109], ne sont pas adaptés à ce changement de perspective. À titre de référence, l'identification de l'intégralité des matrices de vertébrés de Transfac sur le génome humain avec Patser nécessite

```

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

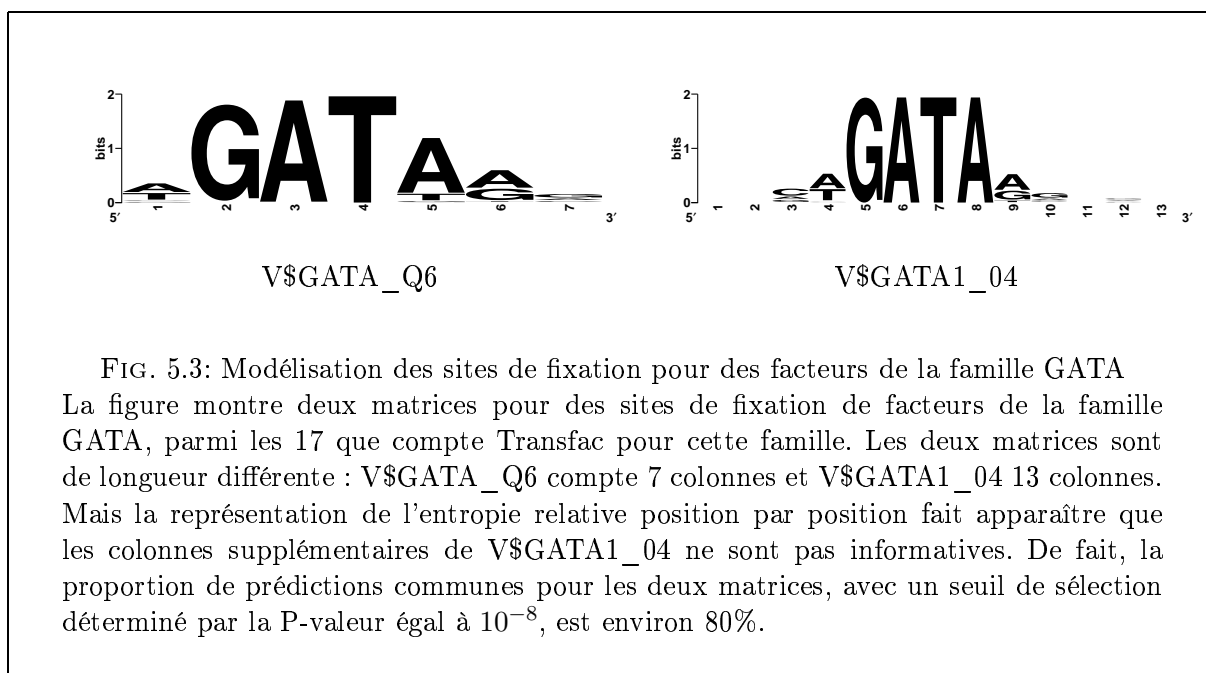
```

A	C	G	T	Position Frequency Matrix				Position Weight Matrix			
7	2	3	3	0.47	0.13	0.2	0.2	0.91	-0.94	-0.32	-0.32
1	14	0	0	0.07	0.93	0	0	-1.8	1.9	-2.3	-2.3
5	9	0	1	0.33	0.6	0	0.07	0.4	1.26	-2.3	-1.8
0	0	15	0	0	0	1	0	-2.3	-2.3	2	-2.3
0	0	15	0	0	0	1	0	-2.3	-2.3	2	-2.3
15	0	0	0	1	0	0	0	2	-2.3	-2.3	-2.3
8	2	0	5	0.53	0.13	0	0.33	1.1	-0.94	-2.3	0.4
4	1	10	0	0.27	0.07	0.67	0	0.11	0.07	1.42	-2.3
1	6	0	8	0.07	0.4	0	0.53	-1.8	0.4	0	1.1
5	4	4	2	0.33	0.27	0.27	0.13	0.4	0.11	0.11	-0.94



FIG. 5.2: Construction d'une matrice de poids pour modéliser le site de fixation du facteur *c-Ets-1*

Le point de départ est un alignement multiple de 15 séquences contenant un site de fixation pour *c-Ets-1* (source : Transfac M00032 [148]). À partir de cet alignement, on construit une matrice de comptage. C'est une matrice de quatre colonnes (une par base) et chaque ligne correspond à une position de l'alignement multiple. Cette matrice est normalisée pour donner la matrice de fréquences. La matrice de poids w est ensuite obtenue en corrigeant par rapport au modèle de fond, avec la formule 5.1. Le calcul suppose ici que les quatre bases sont équiprobables, et il intègre des pseudo-poids pour corriger une éventuelle sur-adaptation. En bas figure la représentation de la matrice de poids sous forme de *logo*, basée sur l'entropie relative (équation 5.2) [120]. Cela fait apparaître les positions conservées et la qualité informationnelle de la matrice de poids.



plus de 24 heures de calcul. L'idée est d'accélérer le calcul en tirant parti du fait que les banques de matrices sont des données stables, que l'on peut pré-traiter. Nous proposons avec TFM-Scan deux types d'optimisation : regrouper les matrices suivant leur similitude et pré-calculer les scores avec une structure d'index.

5.3.1 Groupement de matrices

La première optimisation repose sur une classification des matrices en fonction de la similitude des sites de fixation associés. En effet les matrices ne sont pas toutes indépendantes, et certaines s'organisent naturellement en groupes homogènes. Il y a des arguments biochimiques à cela. Des facteurs de la même famille peuvent avoir des domaines de liaison à l'ADN similaires, et donc des affinités pour des sites de fixation proches. Les facteurs GATA de la figure 5.3 illustrent cette remarque. La proximité de matrices peut également s'expliquer par l'histoire de la constitution de la base de données, avec des données redondantes provenant de différentes sources bibliographiques. Enfin, le regroupement concerne aussi les prédictions issues d'une même matrice, sur le brin direct et sur le brin indirect par exemple avec les sites palindromiques.

Le problème de comparer des matrices a été abordé récemment dans [121]. Nous avons besoin ici d'un mode de comparaison opérationnel qui informe sur les variations de score induites. La comparaison entre matrices se fait suivant un alignement avec insertions et délétions limitées aux colonnes extrémales. Le score d'une substitution entre deux colonnes est donné par la différence maximale de score pour une lettre. Le score des insertions et des délétions est égal au score maximal de la colonne (ce qui correspond à une substitution avec le modèle de fond, décrit par le pourcentage en A, C, G et T). On obtient ainsi une distance qui permet de former des groupements homogènes de matrices. L'idée est ensuite d'associer à chaque groupement une matrice représentante accompagnée d'un intervalle de scores. En dehors de cet intervalle d'incertitude, on peut conclure directement pour chaque matrice du groupe à l'existence ou à la non-existence d'un site. Cette analyse s'appuie sur la distribution des scores des matrices.

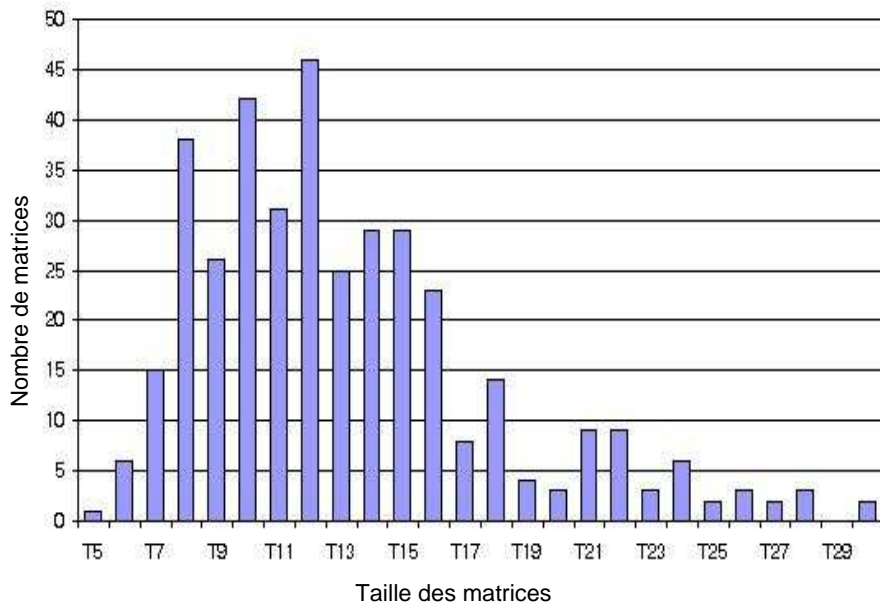


FIG. 5.4: Répartition des longueurs de matrices de vertébrés dans Transfac 6.0

5.3.2 Indexation des matrices

La seconde optimisation est orthogonale à la première, dans le sens où elle s'applique aussi bien aux matrices obtenues par regroupement qu'aux matrices initiales proposées dans les banques. Elle vient du constat suivant : quand on balaye un long texte sur un alphabet fini, *a fortiori* un alphabet de quatre lettres comme l'ADN, il y a nécessairement des répétitions de courts motifs. Cette remarque est à l'origine, de manière beaucoup plus sophistiquée, de l'algorithme des *quatre Russes* pour l'alignement de séquences. Dans le cas des matrices, cela conduit simplement à pré-calculer les scores des matrices pour des motifs de longueur donnée. Ce pré-traitement est d'autant plus efficace que la base de données de matrices est connue à l'avance, et il peut donc être réalisé une fois pour toute.

La longueur moyenne d'une matrice est trop importante pour être traitée par un seul motif. Il est nécessaire de recourir à un index modulaire, avec un découpage qui exploite l'additivité du score. Par exemple, supposons que l'on découpe une matrice de n positions en trois tranches, de longueur i , $j-i$ et $n-j$. Le score final $S(1..n)$ d'un mot est égal à $S(1..i) + S(i+1..j) + S(j+1..n)$. Pour chaque tranche, on pré-cacule l'ensemble des scores possibles, ce qui donne trois index de taille totale égale à $4^i + 4^{j-i} + 4^{n-j}$. Si on applique ce même découpage à toutes les matrices, on obtient un index de taille $(a+b+c)4^i + (b+c)4^{j-i} + c4^{n-j}$, a étant le nombre de matrices de moins de i colonnes, b le nombre de matrices comprenant de $i+1$ à j colonnes et c le nombre de matrices de plus de j colonnes (on suppose que n est la longueur maximale). Il reste à déterminer les paramètres du découpage, c'est-à-dire le nombre et la taille de chacun des index à combiner. Le but est d'optimiser le temps global d'exécution tout en respectant la taille de l'espace disponible. Il n'y a aucun argument *a priori* pour que les différents index soient de

longueur identique. La figure 5.4 montre que la répartition des tailles des matrices n'est pas uniforme. L'idée est de favoriser le changement d'index aux longueurs qui correspondent à des bornes de matrices. Formellement, pour chaque découpage p d'un ensemble de matrices \mathcal{M} , on peut calculer l'espace mémoire $e(p)$ nécessaire à la construction de l'intégralité des structures d'index, et le nombre d'additions $s(p)$ à réaliser pour calculer la totalité des scores à une position donnée dans le texte. L'ensemble \mathcal{M} et l'espace mémoire disponible étant connus, on choisit le découpage p qui minimise $s(p)$ sous la contrainte que $e(p)$ soit inférieure à l'espace disponible. On s'est ainsi ramené à la résolution d'un problème d'optimisation sous contraintes particulièrement simple. Pour l'exemple de la figure 5.4 avec une limite de 256 Mo, on obtient un découpage en quatre index, de taille respective 9, 9, 7 et 5. Le gain de temps par rapport à Paster est alors un facteur 9.

5.4 Régions exceptionnelles

5.4.1 Futilité des prédictions

À la pertinence du modèle des matrices *in vitro*, évoquée en section 5.2, répond toutefois le « théorème » suivant, bien connu des utilisateurs d'outils de prédiction de sites de fixation.

Futility theorem (Wasserman). *Près de 100% des sites de fixation prédits n'ont pas de fonction in vivo.*

Cet état de fait n'est pas tant imputable au modèle des matrices, qu'à la difficulté intrinsèque du problème. Il s'explique par le faible contenu informationnel des séquences de fixation, par la longueur des régions potentiellement régulatrices à considérer. C'est également la conséquence du manque de compréhension des autres mécanismes intervenant dans les modalités de l'interaction ADN-protéines et dans le processus de régulation.

Le problème du bruit se pose avec d'autant plus d'acuité que l'on souhaite travailler à grande échelle. Il est toutefois possible d'améliorer le *futility theorem* en présence de jeux de gènes régulés par un même facteur de transcription, comme c'est le cas typiquement pour l'analyse de données de biopuces. Il est alors possible de travailler plus finement avec la recherche de motifs sur-représentés. L'hypothèse est que les motifs présents dans les régions régulatrices et qui ont une fréquence d'occurrence exceptionnelle sont impliqués dans la régulation de gènes. Cette approche s'est révélée féconde avec la découverte de motifs exacts ([137, 27] pour les procaryotes ou la levure), ou de motifs approchés ([4, 131] pour plusieurs organismes eucaryotes). Il peut également s'agir de la sur-représentation de motifs connus modélisés sous la forme de matrices, comme dans Toucan [1], Mscan [2] ou TFM-Explorer, que nous allons présenter.

Même si le domaine d'application est totalement différent, la démarche relève de la même philosophie que la recherche de structure commune pour les gènes à ARN homologues du chapitre 4. Elle utilise l'évolution pour extraire du signal parmi le bruit. Pour la prédiction de structures d'ARN, il s'agissait de construire une structure secondaire consensus parmi la multitude de structures secondaires potentielles. Ici, on postule que les mécanismes de régulation transcriptionnelle sont conservés pour détecter les sites actifs parmi la multitude de sites prédits.

5.4.2 Sur-représentation locale

TFM-Explorer est une méthode pour extraire dans une famille de séquences des régions présentant des sur-représentations de sites prédits pour un facteur donné dans une famille de

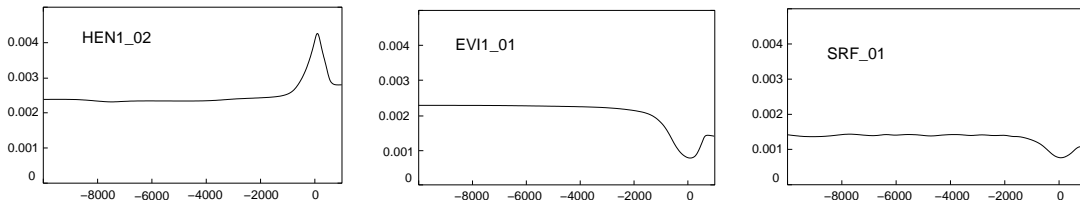


FIG. 5.5: Densité des sites prédits

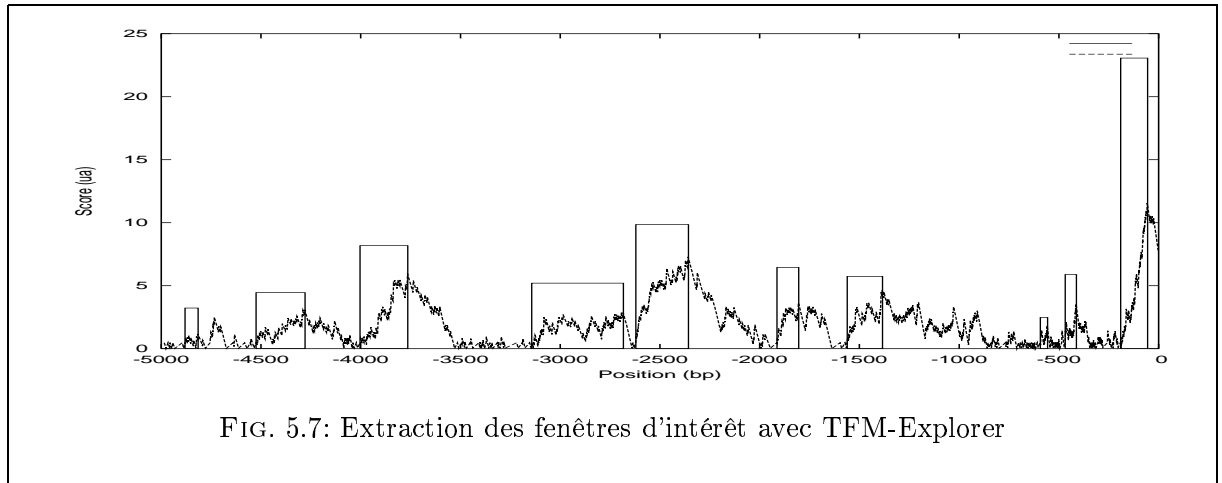
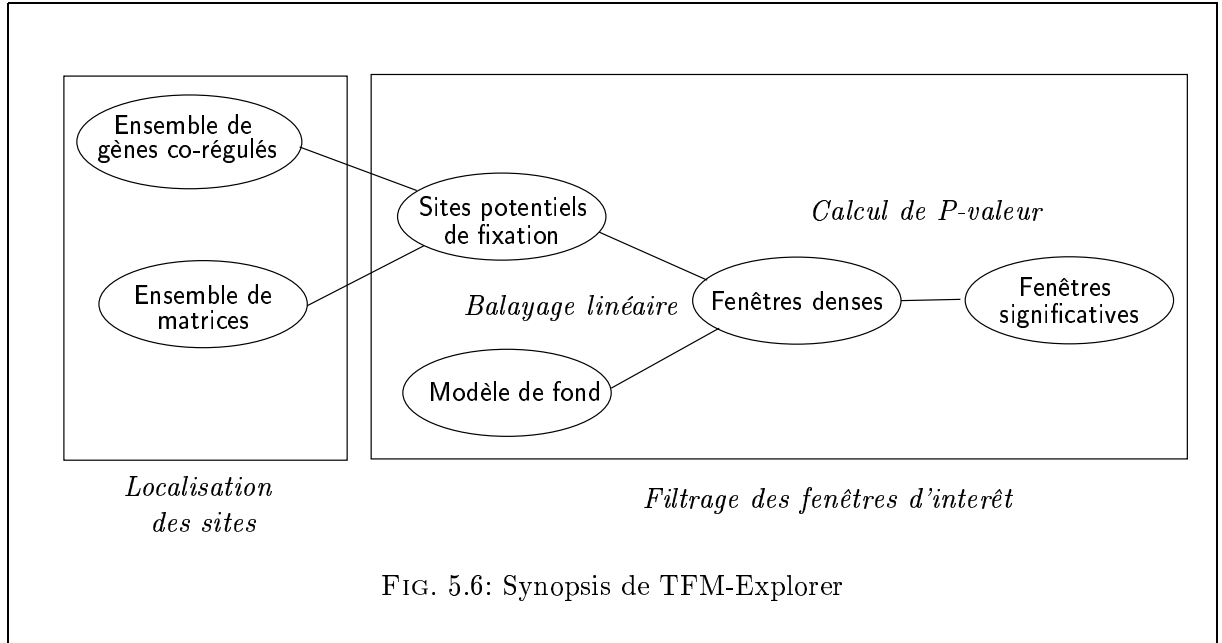
Ces graphiques montrent trois exemples de répartition de sites prédits pour des matrices de composition et de contenu informationnel différents pour des gènes humains. L'axe des abscisses indique la position sur la séquence relativement au site d'initiation de la transcription et l'axe des ordonnées la fréquence d'observation d'un site à cette position.

séquences. Ces idées sont déjà présentes dans [1, 140, 2] qui proposent une analyse statistique de la significativité de tels phénomènes. La nouveauté est d'accompagner le traitement statistique d'une heuristique d'inférence de fenêtres d'intérêt. Aborder le problème sous cet angle offre l'avantage de ne pas avoir à faire d'hypothèses sur la localisation, ni sur la taille des régions recherchées. C'est un point de vue local adapté à l'analyse de longues séquences, où les signaux de régulation ne couvrent qu'une partie des données. Enfin, cela permet de s'affranchir des biais induits par la composition hétérogène des régions potentiellement impliquées dans le contrôle de la régulation, en considérant un modèle de fond positionnel. La figure 5.5 donne trois exemples de distribution de prédiction de sites pour trois facteurs de transcription.

TFM-Explorer intervient après une étape préliminaire de localisation de sites de fixation potentiels. Le déroulement est présenté sur la figure 5.6. Séparer la localisation des sites de leur sélection permet de faire l'économie de l'algorithme de Viterbi. La détection des fenêtres candidates est ensuite basée sur un algorithme de balayage. Le point de départ est la comparaison de deux modèles : le modèle de fond, basé sur l'échantillon de référence, et le modèle cible, décrivant les fenêtres significatives recherchées. Pour chaque position i , ces modèles peuvent être simplement décrits par les paramètres p_i et q_i : p_i est la probabilité d'observer un site à la position i dans l'échantillon de référence et q_i est la probabilité d'observer un site dans une région d'intérêt de l'échantillon d'étude. À chaque position i , on associe un score $s(i)$ qui indique l'appartenance au premier ou au second modèle.

$$s(i) = \log P(q_i, n, k) - \log P(p_i, n, k) \quad (5.3)$$

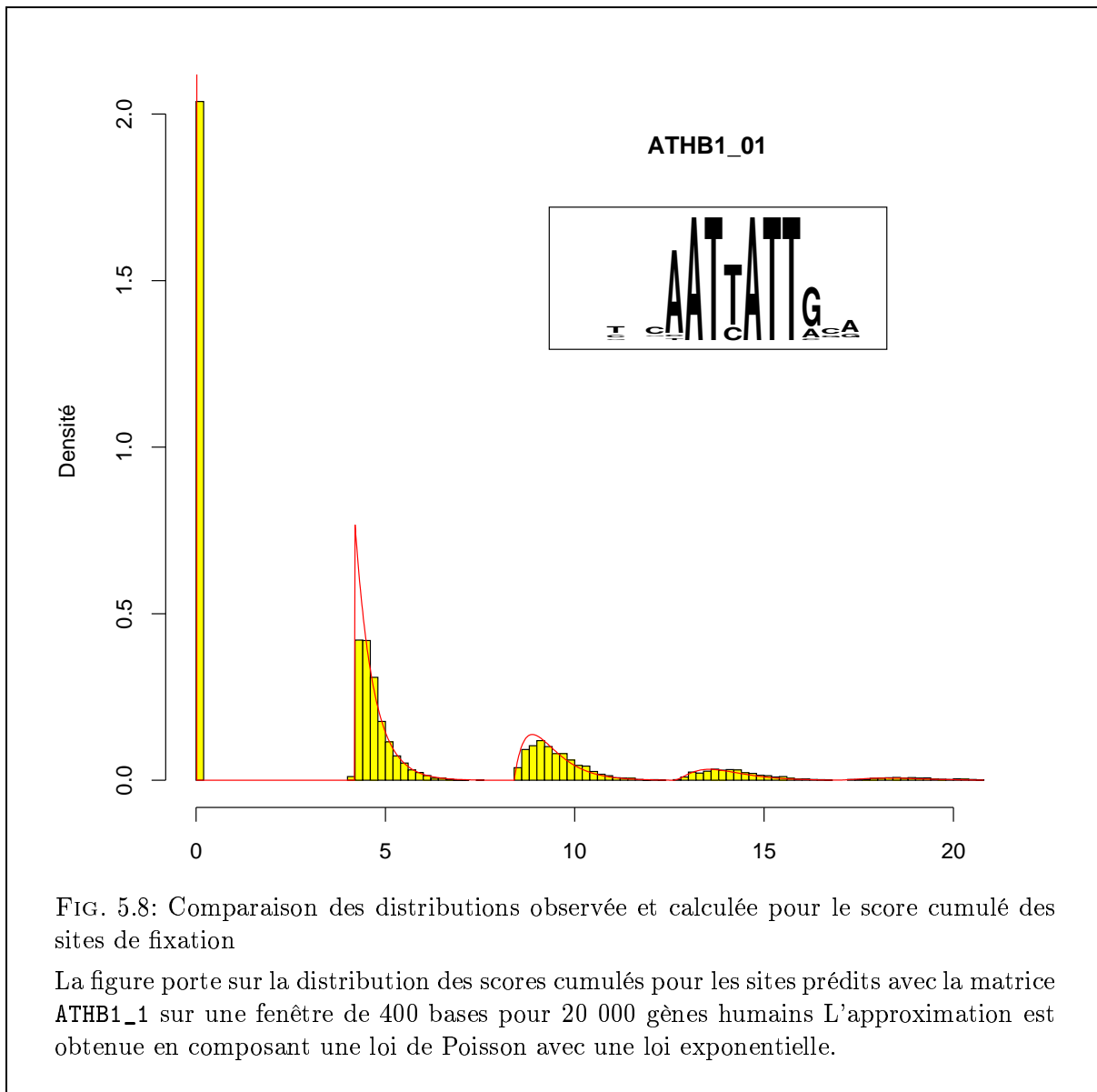
où $P(p_i, n, k)$ est la probabilité d'observer k sites à la position i pour les n séquences dans le modèle de fond, et $P(q_i, n, k)$ la probabilité du même événement dans le modèle cible décrit par q_i . Pour estimer $P(q_i, n, k)$ et $P(p_i, n, k)$, le calcul est simplifié en faisant l'hypothèse que les positions des sites sont indépendantes. Cela est raisonnable, notamment parce que nous ne considérons que les sites non chevauchants et que la distance moyenne entre deux sites est toujours beaucoup plus importante que la longueur d'un site de fixation. La probabilité d'observer k sites à la position i pour les n séquences dans le modèle de fond décrit par p_i se calcule avec une loi binomiale. On obtient ainsi



$$s(i) = k \log \frac{q_i}{p_i} + (n - k) \log \frac{1 - q_i}{1 - p_i} \quad (5.4)$$

Il reste ensuite à déterminer p_i et q_i . Les valeurs de p_i sont naturellement calculées à partir du modèle de fond. Pour q_i , comme nous recherchons les fenêtres avec une forte densité relative, nous appliquons un ratio fixe : $r = \frac{q_i}{p_i}$. Enfin, on extrait les fenêtres candidates en recherchant les positions où le score reste croissant. Pour cela, on utilise un score cumulatif, ayant 0 comme valeur plancher : $S_i = \max(0, S_{i-1} + s_i)$. Ce système est additif par construction. Il est calculé de manière incrémentale le long des séquences (figure 5.7). Les fenêtres ainsi trouvées sont ensuite classées en fonction de leur significativité, avec le calcul de la P-valeur.

La P-valeur d'une fenêtre $[i..j]$ dépend de plusieurs paramètres : les positions i et j sur la séquence, la matrice M , le nombre de sites k , le nombre de séquences n . Cette fois, le raisonnement est sur une fenêtre donnée, et non sur l'ensemble du jeu de séquences, ce qui permet de faire l'hypothèse que la distribution des sites sur cette région est uniforme. La loi de comptage



pour les motifs exacts sans chevauchement est bien étudiée [112]. Sa loi limite est une loi de Poisson, dès lors que l'espérance d'occurrence du motif est petite. Le seuil appliqué aux scores des matrices assure que l'espérance d'occurrence est suffisamment faible. À défaut d'une solution plus satisfaisante, c'est que nous avons retenu. C'est également ce qui est fait dans [140].

Les travaux actuels portent sur l'extension du calcul de la P-valeur pour prendre en compte la qualité des sites de fixation prédits dans le paysage de la régulation. Cela se fait en composant la loi de distribution des sites avec la loi de distribution des scores des matrices. La figure 5.8 montre l'approximation obtenue dans un cas favorable.

TFM-Explorer, dans sa forme actuelle, est accessible publiquement avec un formulaire web⁴.

⁴<http://bioinfo.lifl.fr/TFM-Explorer>

5.4.3 Les facteurs Rel/NF- κ B et leurs gènes cibles

Cette approche a fait l'objet d'une validation, en collaboration étroite avec Karo Gosselin et Corinne Abbadie de l'UMR CNRS 8117 (Institut de Biologie de Lille), sur un jeu de données de gènes régulés par les facteurs de transcription de la famille Rel/NF- κ B. Une première version de ce travail est décrite dans [26].

Les facteurs NF- κ B sont connus pour être impliqués dans le développement de différents types de tumeurs chez l'homme, de par leur participation au contrôle des mécanismes de prolifération, apoptose et sénescence [90]. Ce sont également des régulateurs majeurs des réponses immunitaires et inflammatoires [91]. Chez les vertébrés, cinq protéines de la famille Rel/NF- κ B sont connues : c-Rel, RelA (ou p65), RelB, NF- κ B1 (ou p50) et NF- κ B2 (ou p52). Toutes ces protéines ont en commun dans leur moitié N-terminale un domaine conservé d'environ 300 acides aminés, le Rel Homology Domain (RHD) qui représente le domaine de fixation à l'ADN. Les facteurs Rel/NF- κ B se fixent sur l'ADN sous forme dimérisée en reconnaissant spécifiquement des sites κ B, dont la séquence consensus, un quasi-palindrome, est 5'-GGGRNYYYCC-3' où R est une purine, Y une pyrimidine et N un nucléotide quelconque. Cette affinité est particulièrement élevée (10-10 à 10-13 M pour l'hétérodimère p50/RelA), ce qui distingue cette famille de nombreux autres facteurs de transcription. Bien sûr, l'affinité pour l'ADN des différents dimères varie en fonction de la séquence exacte du motif κ B qui varie d'un promoteur à l'autre. Par exemple, les homodimères p50 ont une meilleure affinité pour les sites parfaitement palindromiques qui permettent la liaison de chaque protéine à un demi-site, tandis que les hétérodimères p50/RelA préfèrent des sites non totalement symétriques [16].

Le jeu de données a été établi par recherche bibliographique d'articles présentant des résultats expérimentaux qui démontrent l'implication d'un ou plusieurs sites κ B dans la régulation de l'expression du gène concerné. La compilation obtenue compte 100 gènes humains [48].

Transfac compte six matrices pour les facteurs Rel/NF- κ B, qui reflètent les spécificités des facteurs de la famille : CREL_01, NFKAPPA50_01, NFKAPPAB65_01, NFKB_Q6, NFKAPPAB_01 et NFKB_C. La recherche a été faite de manière exhaustive pour l'intégralité des matrices de vertébrés de Transfac 6.0, soit 243 matrices. La figure montre les résultats. TFM-Explorer est capable d'extraire des fenêtres de longueur variable, étendue sur plusieurs centaines de bases ou au contraire très brèves, bien localisées.

La forme en diade du site de fixation fait que la détection des sites κ B se prête mal à une analyse par motif exact ou par Gibbs sampling. Nous avons appliqué MotifScanner de la suite Toucan [1], qui fonctionne comme TFM-Explorer en recherchant des sites de fixation produits à l'aide de matrices. Les résultats sont reproduits en Table 5.11. Quand on considère l'intégralité du jeu de données, ou même des régions plus courtes, plusieurs facteurs sont prédits avec une significativité élevée. Mais aucune prédiction ne correspond au résultat attendu. Cela pourrait s'expliquer par la taille de la région et l'utilisation d'un modèle uniforme. Il faut réduire la fenêtre à la région [-500,-1] pour voir apparaître les facteurs de la famille NF- κ B. Les boîtes TATA, quant à elles, ne sont pas détectées. Les autres facteurs prédits sont associés à des matrices riches en GC. On voit là la sensibilité d'une approche locale, par rapport à une approche globale.

Nous avons complété cette étude en l'étendant à la souris. Les gènes orthologues des gènes humains ont été sélectionnés grâce à Homologène [38]. Les résultats obtenus, présentés en figure 5.10, semble indiquer que le contrôle par les facteurs Rel-NF- κ B pour ce jeu de données est conservé chez la souris.

matrice	fenêtre	nbre de sites	nbre de séquences	P-valeur	E-valeur
TATA_01	[-74 : -9]	42	42	3.57e-14	9.55e-05
NFKB_C	[-507 : -16]	204	87	1.71e-13	4.57e-04
NFKAPPAB_01	[-624 : -17]	237	92	3.32e-13	8.87e-04
NFKAPPAB65_01	[-520 : -13]	190	83	6.51e-13	1.74e-03
NFKB_Q6	[-696 : -20]	235	91	1.65e-11	4.41e-02
CREL_01	[-511 : -17]	173	75	1.58e-10	4.22e-01
TATA_C	[-60 : +42]	39	35	3.61e-10	9.65e-01
RREB1_01	[-4382 :-3855]	240	88	7.05e-10	1.88e+00
NFAT_Q6	[-231 : -16]	98	62	6.13e-08	1.64e+02
CDXA_02	[-5849 :-5523]	100	49	1.25e-07	3.34e+02

FIG. 5.9: Résultats de TFM-Explorer pour les gènes cibles de Rel/NF- κ B chez l'Homme. Les séquences promotrices des gènes cibles ont été extraites pour les positions de -10 000 à +1000 à partir de l'assemblage *hg 16* du génome humain proposé par le *Genome Browser* de l'UCSC [74]. Le fait de prendre une longueur importante, 11 000 bases, est volontaire : cela joue le rôle de contrôle négatif. L'ensemble des sites potentiels est déterminé à l'aide des matrices de vertébrés de Transfac 6.0 avec un critère de sélection uniforme donné par la P-valeur. Le modèle de fond est construit à partir de 20.000 séquences promotrices humaines.

Le tableau indique les dix meilleures fenêtres trouvées par TFM-Explorer, toutes matrices confondues. Pour chaque fenêtre, sont précisés le facteur impliqué, la localisation, le nombre total de sites prédits, le nombre de séquences portant ces sites, la P-valeur et la E-valeur. Parmi les six meilleures fenêtres, cinq concernent des matrices Rel/NF- κ B. Les fenêtres ainsi détectées ont une longueur variable, mais avec des positions similaires qui se recoupent dans le promoteur. En les synthétisant, on obtient une région limitée par les positions -520 et -16. Il s'agit donc d'une fenêtre proximale. Cette prédiction est conforme aux informations disponibles concernant les sites connus, puisque sur les 149 sites identifiés expérimentalement que comprend au total la liste des gènes cibles, 74% se trouvent dans la fenêtre [-520, -17]. La localisation des sites résiduels est très variable : certains sont situés plus en amont du site d'initiation de la transcription, jusqu'à -4000, et d'autres après le site d'initiation de la transcription, avant ou après le codon start, dans un exon ou même dans la région flanquante 3'. Enfin, ces fenêtres peuvent comporter plusieurs sites prédits par séquence, ce qui est également cohérent avec les données.

TFM-Explorer détecte également, en première et en huitième positions, une fenêtre avec une sur-représentation importante en boîtes TATA. C'est une particularité intéressante, qui est attendue pour des cibles de facteurs rapidement activables. Les boîtes TATA sont des sites présents en amont des sites d'initiation de la transcription en position -35 à -25. Ils permettent la fixation de facteurs généraux de transcription, dont la TBP et TFIID, qui facilitent le positionnement de l'ARN polymérase II sur le site d'initiation de la transcription. Il est admis que les gènes à boîte TATA sont des gènes facilement et rapidement activables, tandis que les gènes dépourvus de boîte TATA seraient transcrits à des taux plus bas et plus constants [106]. La fenêtre riche en boîtes TATA la plus significative est bien localisée par TFM-Explorer. Elle est délimitée de -59 à -9.

matrice	fenêtre	nbre de sites	nbre de séquences	P-valeur	E-valeur
NFKAPPAB_01	[-726 : -46]	169	70	2.02e-12	5.40e-03
CREL_01	[-757 : -56]	170	62	1.04e-11	2.77e-02
NFKAPPAB65_01	[-892 : -9]	190	66	7.23e-11	1.93e-01
NFKB_Q6	[-735 : -57]	164	64	1.88e-10	5.03e-01
AP2_Q6	[-9431 : -7755]	281	64	1.68e-09	4.50e+00
NFKB_C	[-436 : -19]	118	56	2.58e-09	6.89e+00
TATA_01	[-59 : +230]	45	42	1.86e-08	4.97e+01

FIG. 5.10: Résultats de TFM-Explorer pour les gènes orthologues chez la Souris.

MotifScanner fenêtre [-5000, + 1000]			MotifScanner fenêtre [-500, -1]		
matrice	nbre de sites	P-valeur	matrice	nbre de sites	P-valeur
EVI1_04	151	0.0	SP1_Q6	80	0.0
PBX1_01	124	0.0	NFKAPPAB65_01	50	5.6-13
CEBPA_01	75	0.0	NFKAPPAB_01	47	6.7-13
HFH1_01	95	0.0	NFKB_C	29	2.1-11
HNF3B_01	157	0.0	CREL_01	42	3.2-11
BRN2_01	56	0.0	NFKB_Q6	37	5.3-11
SRY_01	55	0.0	NFKAPPAB50_01	19	5.7-6
ARP1_01	77	0.0	GC_01	52	5.0-4
CEBP_01	88	0.0	IRF2_01	9	0.02

FIG. 5.11: Résultats de MotifScanner pour les gènes cibles humains de NF- κ B
Le modèle de fond est basé sur un modèle de Markov d'ordre 3 construit à partir d'un échantillon d'apprentissage fourni par EPD (ce sont les paramètres recommandés).

5.5 Perspectives

Les travaux présentés dans ce dernier chapitre ne sont pas finalisés. Ils sont en cours de réglage et d'implémentation. Concernant TFM-Explorer, les développements actuels sont une première étape vers la détection de combinaisons de sites de fixation. La recherche avec un modèle local devrait être particulièrement appropriée pour cette évolution. Cela permet de prendre en considération dans une même combinaison des sites distaux et des sites proximaux. Il serait alors possible de dégager des profils d'association de motifs qui pourrait permettre d'identifier des gènes cibles potentiels à l'échelle d'un génome avec TFM-Scan. Ce chapitre est donc riche en perspectives. Mais il est également porteur de plusieurs limites.

En premier lieu, TFM-Scan et TFM-Explorer sont focalisés sur la modélisation de sites de fixation par des matrices. Cette approche est par nature limitée aux protéines régulatrices pour lesquelles les sites ont été identifiés expérimentalement. Un protocole plus satisfaisant serait d'intégrer la découverte de motifs sur-représentés. Des méthodes à base de motifs exacts ou de paires de motifs exacts, pour les domaines de liaison en forme de dimère, ont été testées avec succès sur des données bactériennes et de levure [137, 138]. Pour les organismes eucaryotes supérieurs, les sites de fixation présentent une plus grande variabilité, ce qui suppose de recourir à des motifs approchés. Il existe pour cela plusieurs méthodes heuristiques telles que Consensus [54], MEME [4] ou l'échantillonnage de Gibbs [131], qui reposent sur des stratégies gloutonnes ou itératives de maximisation de la vraisemblance. Il apparaît que ces algorithmes ne sont pas adaptés au traitement de données volumineuses incertaines. En comparaison, les méthodes exactes avec un dénombrement exhaustif présentent des avantages. Elles permettent d'accéder aux solutions sous-optimales et d'éliminer les faux positifs liés aux régions de faible complexité, aux séquences alu, aux répétitions, . . . Pourrait-on s'appuyer sur un modèle de représentation, moins riche que les matrices, qui se prêterait à une résolution algorithmique exacte? Cela a été fait par exemple dans [95], pour des motifs approchés où les erreurs sont distribuées de manière uniforme. Cela ne correspond pas aux sites de fixation connus et répertoriés pour lesquels il apparaît que le contenu en information du motif varie suivant la position dans le site. Nous sommes en train de tester un premier prototype à visées exploratoires qui extrait des séquences consensus comme premier sas de sélection.

La deuxième limite, qui est fondamentale, est le matériel sur lequel nous travaillons. Le problème est de savoir jusqu'à quel point un modèle basé sur la seule analyse des séquences régulatrices peut être informatif. La combinaison de deux, trois voire quatre motifs peut-elle caractériser un gène cible? Dans ce domaine, nous avons le sentiment que la bio-informatique reste à la recherche de modèles physiques nécessaires à la compréhension des mécanismes qui conditionnent l'initiation de la transcription. Cela comprend par exemple la marche du facteur le long de l'ADN, en dimension un ou en dimension trois, avec la possibilité d'existence de sites auxiliaires, et la modélisation de la dynamique de la chromatine. C'est tout un champ de recherches à explorer.

Bibliographie

- [1] S. Aerts, G. Thijs, B. Coessens, M. Staes, Y. Moreau, and B. De Moor. Toucan : deciphering the cis-regulatory logic of coregulated genes. *Nucleic Acids Research*, 31(6) :1753–1764, 2003.
- [2] W.B. Alkema, O. Johansson, J. Lagergren, and W.W. Wasserman. Mscan : identification of functional clusters of transcription factor binding sites. *Nucleic Acids Research*, 32 :W195–8, 2004.
- [3] J. Allali and M.-F. Sagot. Novel tree edit operations for RNA secondary structure comparison. In *WABI. Lecture Notes in Bioinformatics*, 2004.
- [4] T.L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In AAAI Press, editor, *Second International Conference on Intelligent Systems for Molecular Biology*, 1994.
- [5] P. Baldi, S. Brunak, Y. Chauvin, C. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification : an overview. *Bioinformatics*, 16(5) :412–424, 2000.
- [6] G. Bejerano, D. Haussler, and M. Blanchette. Into the heart of darkness : large-scale clustering of human non-coding dna. *Bioinformatics*, 20 (suppl 1) :40–48, 2004.
- [7] M. Bekaert, L. Bidou, A. denise, G. Duchateau-Nguyen, J.P. Forest, C. Froidevaux, I. Hatin, J.P. Rousset, and M. Termier. Towards a computational model for eukariotic -1 frameshifting sites. *Bioinformatics*, 19 :327–335, 2003.
- [8] O.G. Berg and P.H. von Hippel. Selection of DNA binding sites by regulatory proteins. statistical-mechanical theory and application to operators and promoters. *Journal of Molecular Biology*, 193(4) :723–50, 1987.
- [9] B. Billoud, M. Kontic, and A. Viari. Palingol : a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. *Nucleic Acids Research*, 24 (8) :1395–1403, 1996.
- [10] G. Bonfante, A. Cichon, J.Y. Marion, and H. Touzet. Algorithms with polynomial interpretation termination proof. *Journal of Functional Programming*, 11(1) :33–53, 2001.
- [11] J.W. Brown. The Ribonuclease P database. *Nucleic Acids Research*, 27(314), 1999. [http ://www.mbio.ncsu.edu/RNaseP/](http://www.mbio.ncsu.edu/RNaseP/).
- [12] R.E. Bruccoleri and G. Heinrich. An improved algorithm for nucleic acid secondary structure display. *Comput Appl Biosci*, 4 :167–173, 1988.
- [13] W. Buchholz. Proof-theoretic analysis of termination proofs. *Annals of Pure and Applied Logic*, pages 57–65, 1995.
- [14] A.C. Caron. Linear bounded automata and rewrite systems : influence of initial configuration on decision properties. In *CAAP*, volume 493, pages 74–89. Lecture Notes in Computer Science, 1991.
- [15] S. Chawathe. Comparing hierarchical data in external memory. In *Twenty-fifth International Conference on Very Large Data Bases*, pages 90–101, 1999.
- [16] F.E. Chen and G. Ghosh. Regulation of DNA binding by Rel/NF-kappaB transcription factors : structural views. *Oncogene*, 18 :6845–6852, 1999.
- [17] J.H. Chen, S.Y. Le, and J.V. Maizel. Prediction of common secondary structures of RNAs : a genetic algorithm approach. *Nucleic Acids Research*, 28(4) :991–999, 2000.

- [18] D.K.Y. Chiu and T. Kolodziejczak. Inferring consensus structure from nucleic acid sequences. *Computer Applications in the Biosciences*, 7(3) :347–352, 1991.
- [19] E.A. Cichon. Termination orderings and complexity characterisation. *Proof Theory*, 1992.
- [20] E.A. Cichon and P. Lescanne. Polynomial interpretations and the complexity of algorithms. In *CADE'11*, pages 139–147. Lecture Notes in Computer Science, 1992.
- [21] E.A. Cichon and E. Tahhan-Bittar. Ordinal recursive bounds for Higman's theorem. *Theoretical Computer Science*, 201(1-2) :63–84, 1998.
- [22] L.J. Collins, T.J. Macke, and D. Penny. Searching for ncRNAs in eukaryotic genomes : Maximizing biological input with RNAmotif. *Journal of Integrative Bioinformatics*, 2004. journal en ligne, <http://journal.imbio.de/>.
- [23] L.J. Collins, V. Moulton, and D. Penny. Use of RNA secondary structure for studying the evolution of RNase P and RNase MRP. *Journal of Molecular Evolution*, 51(3) :194 – 204, 2000.
- [24] J. Couzin. Breakthrough of the year. small RNAs make big splash. *Science*, 298 :2296–2297, 2002.
- [25] M. Dauchet. Simulation of Turing machines by a left-linear rewrite rule. *Theoretical Computer Science*, 103 :409–420, 1992.
- [26] M. Defrance, H. Touzet, K. Gosselin, and C. Abbadie. Recherche d'éléments régulateurs communs, application aux gènes cibles des facteurs de transcription Rel-kappaB. In *JOBIM*, pages 1–12, 2004. Disponible à <http://bioinfo.lifl.fr/TFM-Explorer>.
- [27] A. Denise, M. Régnier, and M. Vandenbogaert. Assessing the statistical significance of overrepresented oligonucleotides. In *WABI*, volume 2149, pages 85–97. Lecture Notes in Computer Science, 2001.
- [28] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3 :69–116, 1987.
- [29] S. Dulucq and L. Tichit. RNA secondary structure comparison : exact analysis of the zhang-shasha tree edit algorithm. *Theoretical Computer Science*, 306(1-3) :471–484, 2003.
- [30] S. Dulucq and H. Touzet. Analysis of tree edit distance algorithms. In *Combinatorial Pattern Matching*, volume 2676, pages 83–95. Lecture Notes in Computer Science, 2003.
- [31] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.
- [32] S.R. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews Genetics*, 2(12) :919–29, 2001.
- [33] S.R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22 :2079–2088, 1994.
- [34] N. El-Mabrouk and F. Lisacek. Very fast identification of RNA motifs in genomic DNA, application to tRNA search in the yeast genome. *Journal of Molecular Biology*, 264 :46–55, 1996.
- [35] P. Ferraro, L. Tichit, and S. Dulucq. Local similarity between trees. Disponible à <http://www.labri.fr/Person/~tichit/Publications/>, 2003.
- [36] M.C.F. Ferreira and H. Zantema. Total termination of term rewriting. In *Rewriting Techniques and Application*, volume 690, pages 213–227. Lecture Notes in Computer Science, 1993.
- [37] W. Fontana, D.A.M. Konings, P.F. Stadler, and P. Schuster. Statistics of RNA secondary structure. *Biopolymers*, 33 :1389–1404, 1993.
- [38] National Center for Biotechnology Information. Homologene. Disponible à <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=homologene>, 2003.
- [39] G.W. Fox and C.R. Woese. 5S RNA secondary structure. *Nature*, 256(5517), 1975.
- [40] D.N. Frank, C. Adamidi, M.A. Ehringer, C. Pitulle C, and N.R. Pace. Phylogenetic-comparative analysis of the eukaryal ribonuclease P RNA. *RNA*, 6(12) :1895–1904, 2000.
- [41] Z. Galil and R. Giancarlo. Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science*, 64 :107–118, 1989.

- [42] P. Gardner and R. Giegerich. A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatics*, 5(140), 2004.
- [43] M. Garey and D. Johnson. *Computers and Intractability*. Ed. Freeman, 1979.
- [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] A. Geser, A. Middeldorp, E. Ohlebusch, and H. Zantema. Relative undecidability in term rewriting : I. the termination hierarchy. *Information and Computation*, pages 101–131, 2002.
- [46] J. Giesl. Generating polynomial orderings for termination proofs. In *Rewriting Techniques and Application*, volume 914, pages 427–431. Lecture Notes in Computer Science, 1995.
- [47] J. Gorodkin, S.L. Stricklin, and G.D. Stormo. Discovering common stem-loop motifs in unaligned RNA sequences. *Nucleic Acids Research*, 29(10) :2135–2144, 2001.
- [48] K. Gosselin, H. Touzet, and C. Abbadie. Rel/NF-kappaB target genes. Disponible à <http://bioinfo.lifl.fr/NF-KB/>, 2004.
- [49] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy. RFAM : an RNA family database. *Nucleic Acids Research*, 31(1) :439–441, 2003. <http://rfam.wustl.edu/browse.shtml>.
- [50] SE Halford and JF Marko. How do site-specific DNA-binding proteins find their targets? *Nucleic Acids Research*, 32(10) :3040–3452, 2004.
- [51] W.G. Handley and S.S. Wainer. Equational derivation vs. computation. *Annals of Pure and Applied Logic*, 70 :17–49, 1994.
- [52] G.H. Hardy. A theorem concerning the infinite cardinal numbers. *Quarterly Journal of Mathematics*, 35 :87–94, 1904.
- [53] D. Harel and R.E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal of Computing*, 13(2) :338–335, 1984 ?
- [54] G.Z. Hertz and G.D. Stormo. Identification of consensus patterns in unaligned dna and protein sequences : a large-deviation statistical basis for penalizing gaps. *Proceedings of the Third International Conference on Bioinformatics and Genome Research*, pages 201–216, 1995.
- [55] G. Higman. Ordering by divisibility in abstract algebras. *Bull. London Mathematical Society*, 3-2 :326–336, 1952.
- [56] M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz. Local similarity in RNA secondary structures. In *IEEE Bioinformatics Conference*, pages 159–168, 2003. *RNAforester* disponible à <http://bibiserv.techfak.uni-bielefeld.de/rnaforester/>.
- [57] I.L. Hofacker. Vienna rna secondary structure server. *Nucleic Acids Research*, 31(13), 2003. disponible à <http://www.tbi.univie.ac.at/ivo/RNA/>.
- [58] I.L. Hofacker, M. Fekete, and P.F. Stadler. Secondary structure prediction for aligned RNA sequences. *Journal of Molecular Biology*, pages 1059–1066, 2002.
- [59] I.L. Hofacker, W. Fontana, P.F. Stadler, L.S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie*, 125 :167–188, 1994.
- [60] D. Hofbauer. Termination proofs with multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science*, 105-1 :129–140, 1992.
- [61] D. Hofbauer. An upper bound on the derivational complexity of Knuth-Bendix orderings. *Inf. Comput.*, 183(1) :43–56, 2003.
- [62] D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Rewriting Techniques and Application*, volume 355. Lecture Notes in Computer Science, 1988.
- [63] H. Huang, M.C.M. Kao, X. Zhou, J.S. Liu, and W.H. Wong. Determination of local statistical significance of patterns in Markov sequences with application to promoter element identification. *Journal of Computational Biology*, 11(1) :1–14, 2004.
- [64] G. Huet and D.S. Lankford. On the uniform halting problem for term rewriting systems. *Rapport Laboria*, 283, 1978.

- [65] J.A. Jaeger, D.H. Turner, and M. Zuker. Improved predictions of secondary structures for RNA. *PNAS*, 86 :7706–7710, october 1989.
- [66] J. Jansson and A. Lingas. A fast algorithm for optimal alignment between similar ordered trees. *Fundamenta Informaticae*, 56(1,2) :105 – 120, 2003.
- [67] Y. Ji, X. Xu, and G.D. Stormo. A graph theoretical approach for predicting common RNA secondary structure motifs including pseudoknots in unaligned sequences. *Bioinformatics*, 20(10) :1591–1602, 2004.
- [68] T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2) :371–388, 2002.
- [69] T. Jiang, G. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. *Journal of Discrete Algorithms*, pages 257–270, 2004.
- [70] T. Jiang, L. Wang, and K. Zhang. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143(1) :137–148, 1995.
- [71] D.H.J. De Jongh and R. Parikh. Well-partial orderings and hierarchies. *Indagationes Mathematicae*, 14 :195–207, 1977.
- [72] V. Juan and C. Wilson. RNA secondary structure prediction based on free energy and phylogenetic analysis. *Journal of Molecular Biology*, 289 :935–947, 1999.
- [73] S. Kamin and J.J. Lévy. Attempts for generalizing the recursive path orderings. *Université d’Illinois*, 1980.
- [74] D. Karolchik, R. Baertsch, M. Diekhans, TS Furey, A. Hinrichs, Y.T. Lu, K.M. Roskin, M. Schwartz, C.W. Sugnet, D.J. Thomas, R.J. Weber, D. Haussler, and W.J. Kent. The UCSC genome browser database. *Nucleic Acids Research*, 31 :51–54, 2003. <http://genome.ucsc.edu/>.
- [75] S.H. Kim, F.L. Suddath, G.J. Quigley, McPherson, J.L. Sussman, A.H.J. Wang, N.C. Seeman, and A. Rich. Three-dimensional tertiary structure of yeast phenylalanine transfer rna. *Science*, 185 :435–440, 1974.
- [76] S.C. Kleene. *Introduction to Metamathematics*. Wolterns-Noordhof, Gronongen, 1952.
- [77] P. Klein. Computing the edit-distance between unrooted ordered trees. In *6th European Symposium on Algorithms*, pages 91–102, 1998.
- [78] R.J. Klein and S.R. Eddy. Rsearch : finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4(1), 2003.
- [79] B. Knudsen and J. Hein. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15(6) :446–454, 1999.
- [80] B. Knudsen and J. Hein. Pfold : RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Research*, 31(13) :3423–3428, 2003.
- [81] D.E. Knuth and P. B. Bendix. Simple word problems in universal algebras. *Computational Problems in Abstract Algebras*, 1970.
- [82] M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci.*, 36(3) :490–509, 1988.
- [83] J.B. Kruskal. Well-quasi-ordering, the tree theorem and Vázsonnyi’s conjecture. *Trans. American Mathematical Society*, 95 :210–225, 1960.
- [84] S.Y. Le and M. Zuker. Common structures of the 5’ non-coding RNA in enteroviruses and rhinoviruses. *Journal of Molecular Biology*, 216 :729–741, 1990.
- [85] I. Lepper. Derivation lengths and order types of Knuth-Bendix orders. *Theoretical Computer Science*, 269(1-2) :433–450, 2001.
- [86] I. Lepper. Simply terminating rewrite systems with long derivations. *Archive of Mathematical Logic*, 43(1) :1–18, 2004.
- [87] I. Lepper and G. Moser. Why ordinals are good for you. *Heribert Vollmer : First-Order Logic with Groupoidal Quantifiers*. KGS Wien, 2004.

- [88] M. Lescot, P. Déhais, G. Thijs, K. Marchal, Y. Moreau, Y. van de Peer, P. Rouzé, and S. Rombauts. Plantcare, a database of plant cis-acting regulatory elements and a portal to tools for in silico analysis of promoter sequences. *Nucleic Acids Research*, 30(1) :325–327, 2002.
- [89] M. Levitt. Detailed molecular model for transfer ribonucleic acid. *Nature*, 224 :759–763, 1969.
- [90] A. Lin and M. Karin. NF-kappaB in cancer : a marked target. *Semin Cancer Biol*, 13 :107–114, 2003.
- [91] H.C. Liou. Regulation of the immune system by NF-kappaB and IkappaB. *J Biochem Mol Biol*, 35 :537–546, 2002.
- [92] T.M. Lowe and S.R. Eddy. tRNAscan-SE : A program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research*, 25 :955–964, 1997.
- [93] B. Ma, L. Wang, and K. Zhang. Computing similarity between RNA structures. *Theoretical Computer Science*, 276(1-2) :111–132, 2002.
- [94] F. Major, S. Lemieux, and M. Ftouhi. Computer RNA three-dimensional modeling from low-resolution data and multiple-sequence information. *Molecular Modeling of Nucleic Acids*, pages 394–404, 1998.
- [95] L. Marsan and M.-F. Sagot. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *Journal of Computational Biology*, 7(3-4) :345–362, 2000.
- [96] D.H. Mathews, J. Sabina, M. Zuker, and D.H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288 :911–940, 1999.
- [97] D.H. Mathews and D.H. Turner. Dynalign : An algorithm for finding the secondary structure common to two RNA sequences. *Journal of Molecular Biology*, 317 :191–203, 2002.
- [98] F. Mignone, C. Gissi, S. Liuni, and G. Pesole. Untranslated regions of mRNAs. *Genome Biology*, 3 :reviews0004.1–0004.10, 2002. Disponible à <http://genomebiology.com/2002/3/3/reviews/0004>.
- [99] G. Moser and A. Weiermann. Relating derivation lengths with the slow-growing hierarchy directly. In *Rewriting Techniques and Application*, volume 2706, pages 296–310. Lecture Notes in Computer Science, 2003.
- [100] V.L. Murthy and G.D. Rose. RNABase : an annotated database of RNA structures. *Nucleic Acids Research*, 31(1) :502–504, 2003. Disponible à <http://www.rnabase.org/>.
- [101] U. Nagaswamy, N. Voss, Z. Zhang, and G.E. Fox. Database of non-canonical base pairs found in known RNA structures. *Nucleic Acids Research*, 28(1) :375–376, 2000. http://prion.bchs.uh.edu/bp_type/all_record_page_by_type.html.
- [102] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48 :443–453, 1970.
- [103] R. Nussinov and A.B. Jacobson. Fast algorithm for predicting the secondary structure of single stranded RNA. *PNAS*, 77 :6309–6313, 1980.
- [104] B. Onoa and I. Tinoco. RNA folding and unfolding. *Current Opinion in Structural biology*, 14(3) :374–379, 2004.
- [105] A. Ouangraoua and P. Ferraro. Distance multi-échelles entre structures secondaires d’ARN. In *JOBIM 2004*, 2004.
- [106] G. Patikoglou and S.K. Burley. Eukaryotic transcription factor-DNA complexes. *Annu Rev Biophys Biomol Struct*, 26 :289–325, 1997.
- [107] O. Perriquet, H. Touzet, and M. Dauchet. Finding the common structure shared by two homologous RNAs. *Bioinformatics*, 19 :108–116, 2003.
- [108] W. Pohlers. *Proof Theory*, volume 1407. Lecture Notes in Mathematics, Springer-Verlag, 1989.

- [109] K. Quandt, K. Frech, H. Karas, E. Wingender, and T. Werner. Matind and Matinspector - new fast and versatile tools for detection of consensus matches in nucleotide sequence data. *Nucleic Acids Research*, 23 :4878–4884, 1995.
- [110] E. Rivas and S.R. Eddy. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(8), 2001.
- [111] J.W. Robbin. Subrecursive hierarchies. *Ph.D. Princeton*, 1965.
- [112] S. Robin, F. Rodolphe, and S. Schbath. *ADN, mots et modèles*. éditions Belin, collection Échelle, 2003.
- [113] D.A. Rodionov, A.G. Vitreschak, A.A. Mironov, and M.S. Gelfand. Comparative genomics of thiamin biosynthesis in prokaryotes. *Journ. Biol. Chem.*, 277(50), 2002.
- [114] E. Roulet, S. Busso, A.A Camargo, A.J. Simpson, N. Mermoud, and P. Bucher. High-throughput selex sage method for quantitative modeling of transcription-factor binding sites. *Nat Biotechnol.*, 20(8) :831–835, 2002.
- [115] Y. Sakakibara. Pair hidden markov models on tree structures. *ISMB (Supplement of Bioinformatics)*, 19 :232–240, 2003.
- [116] A. Sandelin, W. Alkema, P. Engstrom, W.W. Wasserman, and B. Lenhard. JASPAR : an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Research*, Database issue :D91-4, 2004.
- [117] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45(5) :810–825, 1985.
- [118] D. Sankoff, J.B. Kruskal, S. Mainville, and R.J. Cedergren. Fast algorithms to determine RNA secondary structures containing multiple loops. *Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison*, pages 93–120, 1983.
- [119] D. Schmidt. Well-partial orderings and their maximal order types. *Habilitationsschrift, Fakultät für Mathematik der Ruprecht- Karl-Universität, Heidelberg*, 1977.
- [120] TD. Schneider. Consensus sequence zen. *Applied Bioinformatics*, 1(3) :111–119, 2002.
- [121] E. D. Schones, P. Sumazin, and M.Q. Zhang. Similarity of position frequency matrices for transcription factor binding sites. *Bioinformatics*, 2004. Advance Access published on August 19, 2004.
- [122] P. Schuster and P.F. Stadler. Discrete models of biopolymers. *TBI preprint*, december 1999.
- [123] K. Schütte. *Proof Theory*. Springer-Verlag, 1977.
- [124] B.A. Shapiro. An algorithm for comparing multiple RNA secondary structures. *Computer Applications in the Biosciences*, pages 387–393, 1988.
- [125] R. Staden. Methods for calculating the probabilities of finding patterns in sequences. *Computer Applications in the Biosciences*, 5 :89–96, 1989.
- [126] J. Steinbach. Generating polynomial orderings. *Information Processing Letters*, 49 :85–93, 1994.
- [127] G.D. Stormo. DNA binding sites : representation and discovery. *Bioinformatics*, 16(1) :16–23, 2000.
- [128] G.D. Stormo and D. S. Fields. Specificity, free energy and information content in protein-DNA interactions. *Trends in biochemical sciences*, 23 :109–113, 1998.
- [129] F. Tahi, M. Gouy, and M. Régnier. Automatic RNA secondary structure prediction with a comparative approach. *Computers and Chemistry*, 26(5) :521–530, 2002.
- [130] K.C. Tai. The tree-to-tree correction problem. *Journal of the Association for Comput. Machi.*, 26 :422–433, 1979.
- [131] G. Thijs, K. Marchal, M. Lescot, S. Rombauts, B. DeMoor, P. Rouzé, and Y. Moreau. A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *Journal of Computational Biology*, 9(2) :447–464, 2002.
- [132] I. Tinoco and C. Bustamante. How RNA folds. *Journal of Molecular Biology*, 293 :271–281, 1999.
- [133] H. Touzet. A characterisation of multiply recursive functions with Higman’s lemma. *Information and Computation*, 178 :534–544, 2002.

- [134] H. Touzet. Tree edit distance with gaps. *Information Processing Letters*, 85(3) :123–129, 2003.
- [135] H. Touzet and O. Perriquet. CARNAC : folding families of related RNAs. *Nucleic Acids Research*, 32 (Supplement 2) :142–145, 2004. Disponible à <http://bioinfo.lifl.fr/carnac>.
- [136] F. Tronche, F. Ringeisen, M. Blumenfeld, M. Yaniv, and M. Pontoglio. Analysis of the distribution of binding sites for a tissue-specific transcription factor in the vertebrate genome. *Journal of Molecular Biology*, 266 :231–245, 1997.
- [137] J. van Helden, B. Andre, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotides frequencies. *Journal of Molecular Biology*, 281(5) :827–842, 1998.
- [138] J. van Helden, A.F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Research*, 28 :1808–1818, 2000.
- [139] S. Vialette. On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312(2-3) :223–249, 2004.
- [140] A. Wagner. Genes regulated cooperatively by one or more transcription factors and their identification in whole eukaryotic genomes. *Bioinformatics*, 15(10) :776–784, 1999.
- [141] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21-1 :168 – 173, 1974.
- [142] Z. Wang and K. Zhang. Alignment between two RNA structures. In *Mathematical Foundations of Computer Science*, volume 2136, pages 690–702. Lecture Notes in Computer Science, 2001.
- [143] W.W. Wasserman and A. Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nature Reviews Genetics*, 5(4) :276–287, 2004.
- [144] A. Weiermann. Bounding derivation lengths with functions from the slow growing hierarchy. *preprint Münster*, 1993.
- [145] A. Weiermann. Complexity bounds for some finite forms of Kruskal’s theorem. *Journal of Symbolic Computation*, 18 :463–488, 1994.
- [146] A. Weiermann. Termination proofs by lexicographic path orderings yield multiply recursive derivation lengths. *Theoretical Computer Science*, 139 :355–362, 1995.
- [147] E. Westhof and P. Auffinger. *RNA tertiary structure, Encyclopedia of Analytical Chemistry*, pages 5222–5232. John Wiley & Sons Ltd, Chichester, 2000.
- [148] E. Wingender, X. Chen, R. Hehl, I. Karas, I. Liebich, V. Matys, T. Meinhardt, M. Pruss, I. Reuter, and F. Schacherer. TRANSFAC : an integrated system for gene expression regulation. *Nucleic Acids Research*, 28(1) :316–319, 2000.
- [149] S. Winker, R. Overbeek, C.R. Woese, G.J. Olsen, and N. Pfluger. Structure detection through automated covariance search. *Comput Appl Biosci*, 6(4) :365–371, 1990.
- [150] C.R. Woese, R. Gutell, R. Gupta, and H.F. Noller. Detailed analysis of the higher-order structure of 16S-like ribosomal ribonucleic acids. *Microbiological Reviews*, 47(4) :621–669, 1983.
- [151] C.R. Woese, L.J. Magrum, R. Gupta, R.B. Siegel, D.A. Stahl, J. Kop, N. Crawford, J. Brosius, R. Gutell, J.J. Hogan, and H.F. Noller. Secondary structure model for bacterial 16S ribosomal RNA : phylogenetic, enzymatic and chemical evidence. *Nucleic Acids Research*, 24 :8(10) :2275–2293, 1980. <http://www.rna.icmb.utexas.edu/>.
- [152] C. Workman and A. Krogh. No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. *Nucleic Acids Research*, 27(24), 1999.
- [153] M. Wu and I. Tinoco. RNA folding causes secondary structure rearrangement. *PNAS*, 95 :11555–11560, 1998.
- [154] H. Zantema. Total termination of term rewriting is undecidable. *Journal of Symbolic Computation*, 20 :43–60, 1995.
- [155] H. Zantema. Torpa : termination of rewriting proved automatically. In *Rewriting Techniques and Application*, volume 3091, pages 95 – 104. Lecture Notes in Computer Science, 2004.

- [156] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6) :1245–1262, 1989.
- [157] Z.Hertz and D.Stormo. Identifying DNA and protein patterns with statistically significant alignment of multiple sequences. *Bioinformatics*, 15(7-8) :563–77, 1999.
- [158] M. Zuker. MFOLD web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13) :1–10, 2003.

Annexe administrative

Curriculum vitæ

Hélène Touzet

Laboratoire d'Informatique Fondamentale de Lille, équipe Bioinfo

Adresse professionnelle : *LIFL, Bâtiment M3, USTL, 59 655 Villeneuve d'Ascq cedex*

Courrier électronique : *Helene.Touzet@lifl.fr*

Page personnelle : *http://www.lifl.fr/~touzet/*

1997 : *Doctorat en Informatique*

Université Henri Poincaré – Nancy 1. Mention très honorable avec félicitations

Intitulé : Propriétés combinatoires pour la terminaison de systèmes de réécriture

Directeur : Adam Cichon

1998 : *Maître de conférences* à l'UFR d'IEEA (Informatique, Electronique, Electro-technique et Automatique) de l'Université des Sciences et Technologies de Lille

Encadrement d'étudiants

- 1999 : encadrement de la thèse d'Olivier Perriquet soutenue en 2003 (*Approche algorithmique pour la prédiction de la structure secondaire des ARN*)
- 2002 : co-encadrement (50%) avec Alain Terlutte et Rémi Gilleron (équipe GRAPPA - apprentissage, Université Lille 3) du stage de DEA de Julien Carme (*Application des automates stochastiques à la modélisation d'alignements multiples*)
- 2003 : encadrement du stage de DEA puis de la thèse de Matthieu Defrance (*Algorithmique pour l'analyse des régions régulatrices dans le génomes d'eucaryotes supérieurs*)
- 2004 : co-encadrement (50 %) avec Jean-Stéphane Varré du stage de DEA puis de la thèse de Aude Liefoghe (*Localisation multiple de sites de fixation de facteurs de transcription*)
- 2004 : co-encadrement (70 %) avec Maude Pupin du stage de DEA de Gwenael Monot (*Identification de séquences reporters pour la conception de biopuces*) en collaboration avec la plateforme biopuces de l'Institut Pasteur de Lille.

Vie collective

- Membre nommée du CNU section 27 de 1999 à 2003
- Co-responsable du DESS Bioinfo de l'USTL, devenu master pro, depuis sa création en 2000
- Membre de la Commission de Spécialistes de la section 27 de l'USTL de 2000 à 2003

- Organisation de deux séries de séminaires : *Informatique pour la régulation* en 2002-03 au LIFL (www.lifl.fr/~touzet/regulation.html) et *Transcriptome et Régulation transcriptionnelle* à partir de 2003 au sein de l'IRI (www.lifl.fr/~touzet/transcriptome.html).
- Membre élue du Conseil de Laboratoire du LIFL depuis 2003
- Responsable de l'équipe Bioinfo du LIFL depuis 2003
- Membre du comité de programme de la conférence JOBIM 2004

Logiciels mis à disposition

- CARNAC, prédiction de structures secondaires d'ARN, disponible à bioinfo.lifl.fr/carnac/
- TFM-Explorer, analyse des régions régulatrices, disponible à bioinfo.lifl.fr/TFM-Explorer/

Publications

Revue internationale

1. *Algorithms with Polynomial Interpretation Termination Proof* Guillaume Bonfante, Adam Cichon, Jean-Yves Marion et Hélène Touzet, *Journal of Functional Programming*, 11 (1), p. 33-53, 2001
2. *A Characterisation of Multiply Recursive Functions with Higman's Lemma* Hélène Touzet, *Information and Computation*, 178, p. 534-544, 2002 – version étendue de 11
3. *Tree Edit Distance with Gaps*, Hélène Touzet, *Information Processing Letters*, volume 85 (3), 2003
4. *Finding the common structure shared by two homologous RNAs* Olivier Perriquet, Hélène Touzet et Max Dauchet, *Bioinformatics*, volume 20, p. 108-116, 2003
5. *CARNAC : folding families of related RNAs*, Hélène Touzet et Olivier Perriquet, *Nucleic Acids Research*, volume 32 (Supplement 2), p. 142-145, 2004
6. *Decomposition algorithms for tree editing distance* Serge Dulucq et Hélène Touzet, *Journal of Discrete Algorithms*, à paraître – version étendue de 12

Conférences internationales avec comité de sélection

7. *An Ordinal Calculus for Proving Termination in Term Rewriting* Adam Cichon et Hélène Touzet, actes de CAAP'96 (Trees in Algebra and Programming), *Lecture Notes in Computer Science*, volume 1059, p. 226-240, 1996
8. *Encoding the Hydra Battle as a Rewrite System* Hélène Touzet, actes de Mathematical Foundations of Computer Sciences, *Lecture Notes in Computer Science*, volume 1450, p. 267-276, 1998
9. *A Complex Example of a Simplifying Rewrite System* Hélène Touzet, actes de ICALP'98 (International Colloquium on Automata, Languages and Programming), *Lecture Notes in Computer Science*, volume 1443, p. 507-517, 1998
10. *Complexity Classes and Rewrite Systems with Polynomial Interpretation*, Guillaume Bonfante, Adam Cichon, Jean-Yves Marion et Hélène Touzet, actes de CSL'98 (Computer Science Logic), *Lecture Notes in Computer Science*, volume 1584, p. 372-384, 1998
11. *A Characterisation of Multiply Recursive Functions with Higman's Lemma*, Hélène Touzet, actes de RTA'99 (Rewriting techniques and Applications), *Lecture Notes in Computer Science*, volume 1631, p. 163-174, 1999

12. *Analysis of tree edit distance algorithms*, Serge Dulucq et Hélène Touzet, actes de Combinatorial Pattern Matching, Lecture Notes in Computer Sciences, volume 2676, p. 83-95, 2003

Conférence nationale

13. *Recherche d'éléments régulateurs communs, application aux gènes cibles des facteurs de transcription Rel-kappaB*, Matthieu Defrance, Hélène Touzet, Karo Gosselin et Corinne Abbadie, JOBIM, p. 1-12, 2004

Actions nationales

- Participation aux Actions Spécifiques *Algorithmes et Séquences* et *Modélisation et Algorithmique des Structure d'ARN* du RTP 41 Bioinformatique du département STIC du CNRS en 2003
- BQR USTL 2003 *Rôle des facteurs Rel/NF- κ B dans la sénescence : recherche de cibles transcriptionnelles par étude de transcriptome et analyse bioinformatique*, en partenariat avec le laboratoire Régulation transcriptionnelle au cours de la tumorigénèse mammaire (UMR CNRS 8117 -IBL-Institut Pasteur de Lille-USTL)
- ACI Jeunes Chercheurs 2003 *Algorithmique pour l'analyse de données d'expression géniques*
- Participation à l'ACI ImpBio AReNA 2004 (groupe de travail pluridisciplinaire national sur la structure et la fonction des ARN)