Repliement de l'ARN

Yann Ponty, Vladimir Reinharz

1

Repliement de l'ARN

Yann PONTY[1], Vladimir REINHARZ[2]
[1] *LIX UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, France*
[2] *Département d'Informatique, Université du Québec à Montréal, Canada*

1.1. Introduction

Les Acides RiboNucléiques (ARN) sont des molécules essentielles à tout organisme vivant. Elles sont composées de chaînes de nucléotides Adénine, Cytosine, Guanine et Uracile, représentables par des séquences sur un alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. Lorsque transcrit à partir d'ADN, les Thymines de ce dernier sont remplacées par l'Uracile ($\mathtt{T} \rightarrow \mathtt{U}$). Les molécules fonctionnelles d'ARN couvrent un grand éventail de tailles. Elles peuvent ainsi n'être composées que de 25 nucléotides (nts), dans le cas des micro ARN. De l'autre côté du spectre, certains virus, tels les coronavirus SARS-CoV, ou le HIV, conservent tout leur matériel génétique dans des molécules d'ARN de plus de 30 000 nts.

Cette diversité de longueur reflète une grande diversité fonctionelle. En tant que médiateur, les ARN messagers transportent l'information génétique contenue dans l'ADN vers le ribosome, qui va les décoder afin de synthétiser des protéines. Des ARN sont aussi partie prenante de la machinerie traductionnelle, et impliqués dans la traduction des ARN en protéines. Elles régulent aussi l'expression quantitative des

Repliement de l'ARN,
coordonné par Yann PONTY, Vladimir REINHARZ.

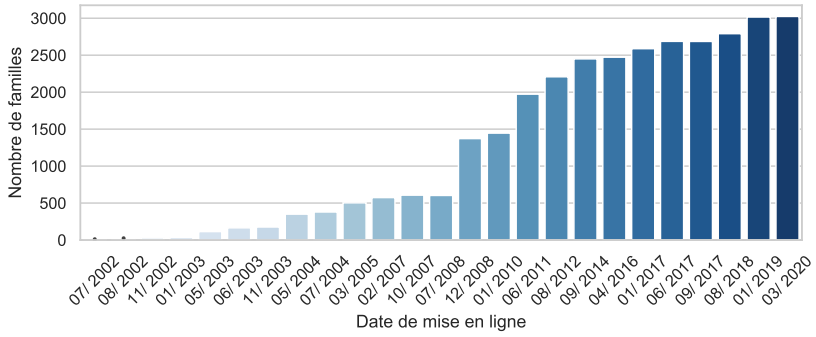


Figure 1.1. *Evolution du nombre de familles fonctionnelles d'ARN, répertoriées au sein de la base RFAM.*

gènes, par exemple via le processus de l'ARN interférence, où de petits ARN simple-brin viennent se fixer sur des ARN messagers. Ils y empêchent alors la fixation du ribosome, et inhibent la synthèse des protéines associées.

Loin d'être exhaustive, cette liste de fonction grandit sans-cesse. La base de données RFAM (Kalvari *et al*. 2017), qui recense et organise en familles fonctionnelles les ARN identifiés dans la littérature, connaît une croissance constante depuis sa création en 2002. Comme l'illustre la figure 1.1, RFAM recense en 2020 plus de 3000 familles fonctionnelles.

1.1.1. *Repliement des ARN*

A la différence des ADN, les ARN sont synthétisés sous la forme d'une copie simple, aussi appelée *simple-brin*, et n'adoptent pas nécessairement une structure en double hélice similaire à l'ADN. Au contraire, l'ARN se replie sur lui-même, soumis à des fluctuations à l'échelle nanométrique. Il se retrouve stabilisé dans certaines de ses conformations, ou structures, par la formation de *paires de bases*, l'appariement de certains de ces nucléotides via la formation de liaisons hydrogènes.

La structure constitue souvent un déterminant essentiel de la fonction au sein des ARN non-codants, non traduits en protéines. Au sein de nombreuses familles fonctionnelles, la structure est davantage conservée au cours de l'évolution que la séquence précise des nucléotides. La prédiction de la structure fonctionnelle d'un ARN représente donc une première étape indispensable pour comprendre son (ou ses) mode(s) d'action(s), et le replacer dans le contexte plus large des systèmes biologiques auxquels ils participent.

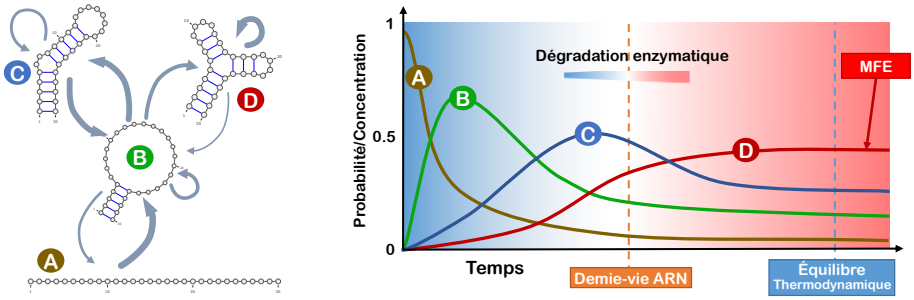


Figure 1.2. *Principaux paradigmes d'étude du repliement.*

1.1.1.1. *Paradigmes pour la prédiction du repliement*

D'un point de vue inspiré par la physique statistique, illustré par la Figure 1.2, l'ARN est initialement transcrit sous une forme essentiellement déstructurée (A). Il fluctue alors de façon stochastique entre ses différentes états, ses *structures* ou *conformations*. Le système finit par atteindre l'*équilibre thermodynamique*, où la probabilité d'observer un ARN dans une conformation donnée cesse d'évoluer avec le temps.

A l'équilibre thermodynamique, l'ensemble des structures adoptables par un ARN ω respecte une *distribution de Boltzmann* basée sur l'énergie libre, où une structure S a pour probabilité

$$\mathbb{P}(S \mid \omega) = \frac{e^{-E(S)/kT}}{\mathcal{Z}}$$

où k est la constante de Boltzmann, $E(S)$ est l'*énergie libre* de S, T la température, et $\mathcal{Z}$ est la *fonction de partition*, une quantité essentielle agissant ici comme une constante de renormalisation. Cette distribution est en effet celle qui maximise l'entropie étant donné l'énergie moyenne, mesurable, du système.

Cette distribution justifie l'accent mis par de nombreuses approches prédictives sur la *structure d'énergie-libre minimale* (Minimum Free-Energy – MFE). En effet, la MFE possède la plus grande probabilité dans la distribution de Boltzmann, et représente donc la structure la plus probablement observée par ses partenaires potentiels dans le milieu cellulaire.

Cependant, bien que maximale, la probabilité de la MFE peut dans l'absolu s'avérer infime, ou simplement minoritaire. En l'absence de pression évolutive, elle est même supposée décroître exponentiellement sur la longueur des ARN considérés. Certaines approches vont donc préférer considérer les propriétés moyennes d'un repliement à l'*équilibre thermodynamique*. Par exemple, dans la Figure 1.2, l'hélice impliquant les deux extrémités de l'ARN est bien plus probable que les deux hélices

internes propre à la MFE. Des approches ensemblistes, basées sur un calcul explicite de la fonction de partition et/ou techniques d'échantillonnage, permettent de calculer ces propriétés moyennes, ainsi que d'éventuelles structures plus représentatives de l'équilibre thermodynamique que la MFE.

Des travaux plus récents s'intéressent aux *propriétés cinétiques* du système, observées avant d'atteindre l'équilibre thermodynamique. En effet, plusieurs phénomènes (repliement co-transcriptionnel, ARN multi-stables) laissent supposer une dépendance en les conditions initiales. Cette-ci vient contredire la notion d'équilibre, et peut être expliquée par des phénomènes de dégradation enzymatiques, empêchant l'ARN de dépasser certaines de ses *barrières d'énergie* en temps comparable avec sa *demie-vie*.

Enfin l'évolution peut aider à la prédiction de la structure fonctionnelle d'un ARN, en postulant l'existence d'une pression de sélection pesant sur la structure des *ARN homologues*, ayant même fonction. En renversant le point de vue, une collection d'ARN homologues devient susceptible d'adopter collectivement une structure commune, induisant un ensemble de contraintes supplémentaires qui vient compléter l'hypothèse de stabilité thermodynamique.

1.1.2. *La structure secondaire*

Il est généralement admis que le processus du repliement des ARN s'effectue de façon hiérarchique. Initialement, l'ARN se replie sur lui-même, établissant un ensemble d'appariements canoniques sans croisement, donnant lieu à une structure arborescente appelée la *structure secondaire*. Dans un deuxième temps, la structure tridimensionnelle adoptée par l'ARN rend possible la formation de motifs plus faiblement stabilisateurs.

Parmi ceux-ci, on trouve des liaisons non-canoniques et des motifs topologiques complexes appelés pseudo-noeuds, constitués de couples d'appariements en situation de croisement quand dessinés sur le demi-plan supérieur. La prédiction de la forme adoptée par un ARN à l'issu de son repliement commence donc souvent par la prédiction d'une (ou plusieurs) structure(s) secondaire(s) candidate(s) pour celui-ci. Cette prédiction initiale est ensuite complétée par la modélisation des éléments supplémentaires, et enfin par l'agencement tri-dimensionnel de ces motifs.

Formellement, une structure secondaire est un ensemble $S \subset [1,n]^2$ de paires de bases $(i,j), 1 \leq i < j \leq n$, satisfaisant les contraintes suivantes :

1) Distance minimale θ : Si $(i,j) \in S$, alors on a $j - i > \theta$.

2) Monogamie : Toute position est impliquée dans *au plus* une paire de S.

3) Croisements interdits : Si $(i,j),(k,l) \in S$ telles que $i < k$, alors on a
$$i < k < l < j \text{ ou } i < j < k < l.$$

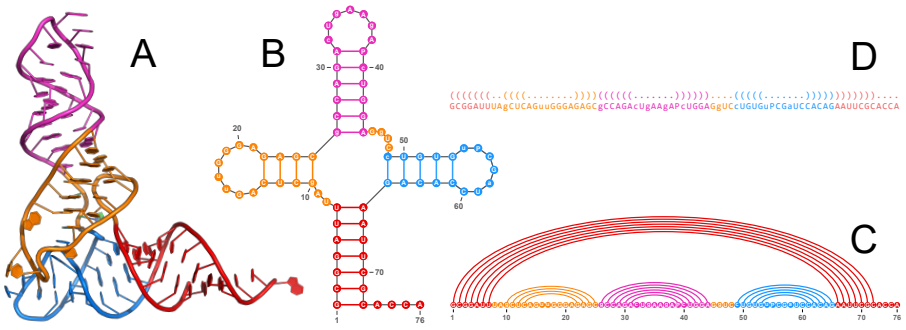


Figure 1.3. *Représentation de la structure secondaire associée au repliement 3D d'un ARN de transfert (PDB 1EHZ, chaîne A)*

Sous ses restrictions, on peut représenter une structure secondaire sous plusieurs formes, comme l'illustre la Figure 1.3. A partir d'une structure 3D d'ARN de la Protein Data Bank (PDB, A), une structure secondaire peut être extraite via l'outil `DSSR` (Lu *et al*. 2015). Celle-ci peut ensuite être dessinée sans croisement comme un graphe planaire extérieur (B), une séquence arc-annotée (C), ou encore une séquence bien parenthésée t sur l'alphabet $\{(,),\bullet\}^n$ (D). Dans cette dernière représentation, toute paire de base (i,j) est matérialisée par une paire de parenthèses $(t_i,t_j)=((,))$ en correspondance, alors que toute position libre (ou non-appariée) est indiquée par un caractère $t_i=\bullet$.

Décrivons enfin l'ensemble des structures possiblement adoptées par une séquence d'ARN $\omega\in\{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{U}\}^n$. Une structure secondaire S est *compatible* avec ω si toute paire de bases $(i,j)\in S$ est *canonique*, c'est à dire que

$$(\omega_i,\omega_j)\in\{(\mathtt{G},\mathtt{C}),(\mathtt{C},\mathtt{G}),(\mathtt{A},\mathtt{U}),(\mathtt{U},\mathtt{A}),(\mathtt{G},\mathtt{U}),(\mathtt{U},\mathtt{G})\}\,.$$

On dénotera par $\mathcal{S}_\omega$ (ou simplement $\mathcal{S}$ quand le contexte le permet) l'ensemble des structures compatibles avec l'ARN ω, et par $\mathcal{S}_{i,j}$ les structures secondaires compatibles avec la région $[i,j]$ de ω.

1.1.2.1. *Modèle d'énergie et décomposition de l'espace des structures*

La stabilité d'un ARN est physiquement déterminée par son énergie libre, exprimée en kcal.mol^{-1}. Plus celle-ci est basse, plus la molécule d'ARN est stable. L'énergie libre d'une structure dépend en grande partie de ses paires de bases, et de l'interaction de celles-ci sous la forme de motifs stabilisant la structure de l'ARN.

Afin d'illustrer les différentes approches algorithmiques disponibles pour la prédiction de structure, nous allons considérer un *modèle d'énergie simple*, défini

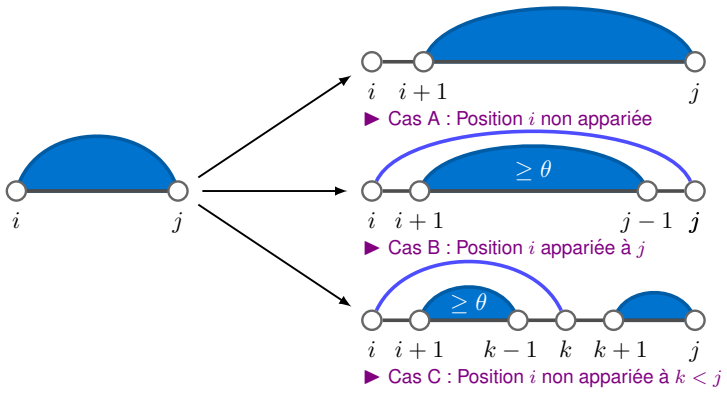


Figure 1.4. *Décomposition de l'espace des structures secondaires, adoptables par un ARN sur une région $[i, j], 1 \leq i < j \leq n$, pour une distance minimale θ entre deux positions appariées.*

additivement sur les paires de bases. Plus précisément, soit S une structure secondaire pour une séquence ω, on a :

$$E(S) := \sum_{(i,j)\in S} E^{\omega}_{i,j}$$

où $E^{\omega}_{i,j}$ est la différence d'énergie associée à la création de la paire (i, j).

On pourra associer aux paires de base canoniques (`G-C`, `A-U`, et `G-U`) une énergie de -1, et à toutes les autres une énergie de $+\infty$, faisant ainsi coincider la minimisation d'énergie avec la maximisation des paires canoniques. Alternativement, on pourra considérer un modèle d'énergie favorisant les paires de bases réputées plus stables (`G-C` $\rightarrow$ -3, `A-U` $\rightarrow$ -2), au détriment de celles plus transientes (`G-U` $\rightarrow$ -3).

Comme illustré à la Figure 1.4, il est possible de décomposer les structures secondaires de $\mathcal{S}_{i,j}$, compatibles avec une région $[i, j]$ d'un ARN ω. Pour cela, on considère le statut du nucléotide i dans une structure formée sur $[i, j]$:

Cas A Soit i libre, et suivie par une structure secondaire formée indépendemment sur la région $[i + 1, j]$;

Cas B Soit i est apparié à la position j, $j - i > \theta$, et alors il se forme une structure secondaire sur la région $[i + 1, j - 1]$;

Cas C Soit i est apparié à une position $k < j$, $k - i > \theta$, et des structures se forment alors dans les régions $[i + 1, k - 1]$ et $[k + 1, j]$. Celles-ci sont indépendantes, du fait de l'interdiction des croisements.

On peut montrer que cette décomposition est *complète*, c'est à dire que toute structure de $\mathcal{S}_{i,j}$ est engendrée/décomposée par l'un des trois cas ci-dessus. De plus, elle est *non-ambiguë*, et toute structure de $\mathcal{S}_{i,j}$ peut être engendrée/décomposée de façon unique en appliquant récursivement les trois cas. Enfin, elle est *correcte* au regard de notre modèle d'énergie simple, car elle fait apparaître explicitement les créations de paires de bases, permettant ainsi une prise en compte du modèle d'énergie.

1.2. Optimisation pour la prédiction de structure

1.2.1. *Produire la structure d'énergie minimale*

La *structure d'énergie minimale* représente la structure la plus stable parmi l'ensemble des structures adoptées par une séquence. Elle est aussi la plus probable à l'équilibre thermodynamique et, à ce titre, représente une candidate raisonnable dans la recherche d'une conformation fonctionnelle pour un ARN donné.

Cependant, sa détermination se heurte à l'explosion combinatoire du nombre de structures secondaires, en nombre attendu $\sim 1.8^n$ (Zuker and Sankoff 1984) pour une séquence d'ARN de longueur n. Le calcul d'une structure d'énergie minimale représente donc un problème d'optimisation combinatoire potentiellement difficile, qu'on définit comme suit.

Minimisation de l'énergie libre

Entrée: Séquence $\omega \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}^+$, $|\omega| = n$.
Sortie: Structure secondaire $S^\star$ telle que

$$E(S^\star) = \min_{S \in \mathcal{S}} E(S)$$

Un schéma de programmation dynamique pour ce problème s'appuie sur la décomposition introduite en section 1.1.2.1.

Dans ce schéma, nous nous intéressons à l'énergie minimale $m_{i,j}$ accessible par un repliement sur une région $[i, j]$ d'un ARN. Dans toute structure sur $[i, j]$, et a fortiori pour toute structure d'énergie minimale, il n'existe que trois options possibles pour la position i : Soit i est libre, et l'énergie minimale d'une structure est trouvée en optimisant l'énergie sur $[i + 1, j]$; Soit i est apparié avec j, et il se forme sur $[i + 1, j]$ un repliement optimal ; Soit i est apparié à $k < j$, et il se forme alors deux repliement, optimaux et indépendants, sur $[i + 1, k - 1]$ et $[k + 1, j]$.

ALGORITHME 1 : Remplissage de la matrice des énergies minimales

Entrée : ω – ARN de taille n
Sortie : m – Matrice m, remplie selon [1.1]

Fonction `DescenteMin` (ω)**:**
  $m \leftarrow$ `MatriceVide`$(n \times n)$
  // Initialiser à 0 toutes les valeurs jusqu'à θ de la diagonale.
  pour $i \leftarrow 1$ **à** n **faire**
    pour $j \leftarrow i$ **à** $\min(i+\theta, n)$ **faire**
      $m_{i,j} \leftarrow 0$
  pour $i \leftarrow n$ **à** 1 **faire**
    pour $j \leftarrow i+\theta+1$ **à** n **faire**
      ▶ Cas A : Position i laissée sans partenaire
      $m_{i,j} \leftarrow m_{i+1,j}$
      ▶ Cas B : Positions i et j forment une paire de base
      $m_{i,j} \leftarrow \min\left(m_{i,j}, m_{i+1,j-1} + E^{\omega}_{i,j}\right)$
      ▶ Cas C : Position i appariée à $k < j$
      pour $k \leftarrow i+\theta+1$ **à** $j-1$ **faire**
        $m_{i,j} \leftarrow \min\left(m_{i,j}, m_{i+1,k-1} + m_{k+1,j} + E^{\omega}_{i,k}\right)$
  retourner m

Pour une région $[i, j]$ telle que $j - i > \theta$, le système à minimiser est formellement décrit par l'équation suivante :

$$m_{i,j} = \min \begin{cases} m_{i+1,j} & \blacktriangleright \text{ A : Pos. } i \text{ est libre} \\ E^{\omega}_{i,j} + m_{i+1,j-1} & \blacktriangleright \text{ B : Paire } (i,j) \\ \min\limits_{k=i+\theta+1}^{j-1} E^{\omega}_{i,k} + m_{i,k-1} + m_{k+1,j} & \blacktriangleright \text{ C : Paire } (i,k), k > j \end{cases} \quad [1.1]$$

Quand $j - i \leq \theta$ on a $m_{i,j} = 0$ car la région est alors trop petite pour contenir une paire de base.

En pratique, nous avons besoin de deux algorithmes, de structures assez similaires, pour produire un repliement d'énergie minimale, et ainsi résoudre ce problème :

– L'algorithme 1 calculera l'énergie minimale associée à chaque région $[i, j]$ dans la séquence, selon le système [1.1].

– L'algorithme 2 reconstruira une structure $S^{\star}$ d'énergie minimale pour la séquence ω.

1.2.1.1. *Correction des algorithmes*

Proposition 1. *L'algorithme 1 retourne une matrice m qui contient à la position $m_{i,j}$ l'énergie minimale de la sous-séquence $\omega_{i,j}$.*

ALGORITHME 2 : Remontée pour la maximisation de paires de bases.

Entrée : $[i, j]$ – Région considérée
m – Matrice de prog. dyn., préalablement calculée selon l'Equation [1.1]
ω – ARN de taille n
Sortie : $S^\star$ – Structure minimisant l'énergie libre

1 **Fonction** `RemontéeMin` (i, j, m, ω)**:**
2 **si** $j - i \leq \theta$ **alors**
3 **retourner** $\overbrace{\bullet \ldots \bullet}^{j-i+1}$ // La structure vide est seule, et a énergie minimale
4 **sinon**
 ▶ Cas A : Position i laissée sans partenaire
5 **si** $m_{i,j} = m_{i+1,j}$ **alors** // Energie min. réalisée par struct. où i est libre
6 $S_i^\star \leftarrow$ `RemontéeMin`$(i + 1, j, m, \omega)$
7 **retourner** $\bullet\, S_i^\star$
 ▶ Cas B : Positions i et j forment une paire de base
8 **si** $m_{i,j} = m_{i+1,j-1} + E^\omega_{i,j}$ **alors** // Energie min. réalisée par struct. appariant i et j
9 $S_{i,j}^\star \leftarrow$ `RemontéeMin`$(i + 1, j - 1, m, \omega)$
10 **retourner (** $S_{i,j}^\star$ **)**
 ▶ Cas C : Position i appariée à $k < j$
11 **pour** $k \leftarrow i + \theta + 1$ **à** $j - 1$ **faire**
12 **si** $m_{i,j} = m_{i+1,k-1} + m_{k+1,j} + E^\omega_{i,k}$ **alors** // Struct. optimale apparie i et k
13 $S_1^\star \leftarrow$ `RemontéeMin`$(i + 1, k - 1, m, \omega)$
14 $S_2^\star \leftarrow$ `RemontéeMin`$(k + 1, j, m, \omega)$
15 **retourner (** $S_1^\star$ **)** $S_2^\star$

Démonstration. Nous allons montrer directement que, à chaque étape du calcul, la valeur obtenue pour $m_{i,j}$ est correcte. Premièrement, comme indiqué à la Section 1.1.2, il ne peut y avoir de paire de base sur une région de longueur plus petite que $\theta + 2$. On peut donc initialiser $m_{i,j}$ avec 0 pour toute les régions $[i, j]$ telles que $j - i + 1 < \theta + 2$, équivalent à $j - i \leq \theta$, ce que réalise la double boucle commençant à la ligne 3.

Maintenant, dans la boucle commençant à la ligne 6, nous remplissons la matrice m une case à la fois. Nous itérons sur les régions $[i, j]$ par ordre croissant sur i (et ensuite sur j). Nous pouvons ainsi garantir que, au moment de calculer un $[i, j]$ précis, les valeurs $m_{i',j'}$ tels que $i' < i$ ont déjà été calculées.

Nous allons donc supposer correctes ces valeurs $m_{i',j'}$, $i < i'$ et montrer que cela implique la correction de $m_{i,j}$. Afin de calculer la valeur de $m_{i,j}$ il y a trois cas possibles :

Cas A Une structure optimale laisse la position i sans partenaire. La structure optimale est composée d'un repliement indépendant sur $[i + 1, j]$, et on trouve donc son énergie en $\omega_{i+1,j}$, déjà correctement calculé.

Cas B Une structure optimale apparie les positions i et j . Ce repliement est donc constitué d'une paire (i, j), d'énergie $E^{\omega}_{i,j}$, et d'un repliement optimal sur $[i + 1, j - 1]$, d'énergie trouvée en $m_{i+1,j-1}$. Ce dernier peut être supposé calculé correctement car $i + 1 > i$.

Cas C Une structure optimale apparie les positions i et $k < j$. La paire (i, k) délimite donc des régions $[i + 1, k - 1]$ et $[k + 1, j]$, où l'ARN forme des repliements indépendants (interdiction des croisements). Ces deux régions démarrent après i, et les énergies minimales sur ceux-ci peuvent donc être trouvées en $m_{i+1,k-1}$ et $m_{k+1,j}$. La somme de ces termes est donc complétée par la contribution $E^{\omega}_{i,k}$ de la paire pour obtenir l'énergie minimale.

Comme toute structure tombe dans un de ces catégorie, la valeur assignée à $m_{i,j}$ représente bien l'énergie minimale d'une structure sur $[i, j]$. On en déduit donc sur la correction de $m_{i,j}$ et, par induction, la correction du calcul sur toute région. □

Proposition 2. *Soit la matrice m calculée par l'algorithme 1, la fonction* `RemontéeMin`(i, j, m, ω) *de l'algorithme 2 retourne une structure d'énergie minimale sur la région* $[i, j]$.

Démonstration. Par la proposition 1 nous savons que $m_{i,j}$ contient l'énergie minimale pour toute région $[i, j]$. Il y a alors quatre cas possibles pour la structure optimale :

Cas 0 : Séquence est trop courte $i - j < \theta$. Nous savons que si la séquence contient moins de θ nucléotide, elle ne peut former de paire de base. La structure sans paire de base, retournée à la ligne 2 est donc d'énergie minimale.

Cas A : Position i laissée sans partenaire. On a alors $m_{i,j} = m_{i+1,j}$. La structure qui commence par une position libre, et forme une structure optimale sur $[i + 1, j]$, a alors énergie $m_{i+1,j}$, et est donc aussi d'énergie minimale pour $[i, j]$.

Cas B : Positions i et j appariées. On a alors $m_{i,j} = m_{i+1,j-1} + E^{\omega}_{i,j}$, coincidant avec l'énergie de la structure appariant i à j, et formant un repliement optimal sur $[i + 1, j - 1]$, que renvoie l'algorithme.

Cas C : Position i appariée à $k < j$. On a alors $m_{i,j} = m_{i+1,k-1} + m_{k+1,j} + E^{\omega}_{i,k}$. Or, cette énergie est bien celle de la structure, renvoyée par l'algorithme, contenant la paire (i, k), et deux repliements sur $[i + 1, k - 1]$ et $[k + 1, j]$.

Comme les cas ci-dessous couvrent toutes les structures possibles, on en conclue que la structure renvoyée à bien énergie optimale sur sa région. □

1.2.1.2. *Analyse de complexité*

La complexité générale de l'algorithme ci-dessus pour la production d'une structure secondaire d'énergie minimale est de $\Theta(n^3)$ en temps, et $\Theta(n^2)$ en mémoire.

Pour `DescenteMin`, après la création de la matrice m en $\Theta(n^2)$, qui borne la complexité en mémoire, l'initialisation prend un temps $\Theta(n)$, la boucle interne ayant un nombre d'itération borné par une constante θ. La principale contribution à la complexité est due aux trois boucles **pour** imbriquées (ligne 6 et au delà). Les deux premières boucles énumèrent toutes les régions $[i, j]$ telles que $j - i > \theta$, et la dernière place choisit $k \in [i + \theta + 1, j - 1]$. Chacune de ces boucles est executée au plus n fois, et la complexité en temps est en $\mathcal{O}(n^3)$, *i.e.* asymptotiquement bornée par $C.n^3$ où C est une constante.

Pour prouver l'équivalence asymptotique, et donc la complexité en $\Theta(n^3)$, on peut raisonner sur les triplets (i, k, j) associés aux exécutions de la boucle la plus intérieure (ligne 10). On remarque que ceux-ci correspondent aux façon de choisir 3 éléments distincts, sans ordre, parmi $n - C'$, où C' est une constante de n :

$$\binom{n - C'}{3} = \frac{(n - C')(n - C' - 1)(n - C' - 2)}{3!} = \frac{n^3}{6} + \mathcal{O}(n^2) \in \Theta(n^3).$$

Pour déterminer la complexité de `RemontéeMin`, on remarque que, en excluant les appels récursifs, le nombre d'opérations réalisées par l'algorithme est linéaire sur la taille de la région $[i, j]$. De plus, les appels récursifs portent sur des sous-régions dont la taille cumulée est décroissante. Il s'ensuit que, dans l'arbre des appels récursifs, le nombre total d'itération de la boucle la plus interne, sommé sur tous les appels à profondeur p, reste bornée par n. Comme la taille des régions est strictement décroissante lors des appels récursifs successifs, la profondeur de l'arbre est bornée par n. La complexité au pire est alors en $\Theta(n^2)$, et la complexité de `RemontéeMin` est toujours dominée par celle de `DescenteMin`.

1.2.1.3. *Pour aller plus loin*

Malgré sa simplicité, ce modèle produit déjà des prédictions informatives, comme on peut le voir à la Section 1.4.2. Elles peuvent être assez largement améliorées en considérant un modèle énergétique plus réaliste (voir Section 1.4.1). Celui-ci nécessite alors un algorithme plus complexe, mais de principe très similaire à celui présenté ici.

Bien que l'algorithme soit communément attribué à Nussinov *et al*. (1978), la version proposée était alors ambiguë et, quoique correcte pour la minimisation, et interdisait tout calcul de la fonction de partition introduite en Section 1.3.1. La version présentée ici est inspirée de travaux combinatoire antérieurs de Waterman (1978).

L'algorithme peut être utilisé pour *prédire l'interaction de deux ARN*. Il suffit pour cela d'exécuter l'algorithme sur la concaténation de deux ARN, intercalées avec θ nucleotides anonymisés (`N`) pour permettre l'appariement des extrémités respectives des

ARN. On obtient alors un complexe d'énergie minimale, composé à la fois de paires de bases internes aux deux ARN, et d'interactions, *i.e.* de paires de bases impliquant les deux brins.

Cependant, l'interdiction des croisements limite alors les interactions à des positions dans la face extérieure de chacun des deux ARN. Des schémas de programmation dynamique plus sophistiqués (Mückstein *et al*. 2006) devront donc être utilisés pour permettre l'exploration d'espace de conformations plus réalistes, permettant par exemple l'interaction des boucles.

Il peut aussi être utilisé pour *simplifier* une structure $S^{\pitchfork}$ présentant des pseudo-noeuds, de façon à obtenir une structure secondaire. Pour cela, on postulera un modèle d'énergie où $E^{\omega}_{i,j} := -1$ si $(i,j) \in S^{\pitchfork}$, et $E^{\omega}_{i,j} := +\infty$ sinon. Minimisation d'énergie revient alors à maximiser le nombre de paires de $S^{\pitchfork}$, et évite ainsi les croisements tout en maximisant, en un sens, l'information structurale résiduelle.

1.2.2. *Lister les repliements sous-optimaux*

Le paradigme de minimisation d'énergie, quoique fécond, reste très sensible au choix d'un modèle d'énergie. En particulier, il est potentiellement impacté par d'éventuelles erreurs de mesures dans les contributions individuelles du modèle d'énergie. Par exemple, il peut arrive qu'une structure S soit jugée plus stable qu'une structure S', et donc renvoyée par un algorithme de minimisation d'énergie, alors que la distance $|E(S') - E(S)|$ est arbitrairement petite, bien plus petite que l'imprécision expérimentale des protocoles utilisés pour calibrer le modèle d'énergie.

Dans un tel cas, il semble arbitraire de ne proposer que la structure S comme représentative du repliement, a fortiori quand les structures quasi-optimales s'avèrent significativement différentes. Ceci motive la prise en compte des *structures (sous-)optimales Δ-admissibles*, situées à distance au plus Δ kcal.mol^{-1} de la structure d'énergie libre minimale. Ce problème est initialement considéré par Zuker (1989), dans une version restreinte à des ensembles de structures n'ayant, deux à deux, aucune paire de base en commun.

Cependant, cette solution est très dépendante de l'ordre de production des structures, et peut ignorer certaines structures stables à cause des ses choix antérieurs. Une version plus satisfaisante, exhaustive, du problème est alors considérée par Wuchty *et al*. (1999).

Repliements Δ-sous optimaux

Entrée: Séquence $\omega \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{U}\}^+$; Tolérance $\Delta \in \mathbb{R}^+$.
Sortie: Ensemble $\mathcal{S}_\Delta$ de structures secondaires Δ-admissibles, c'est à dire ayant

une énergie libre distante d'au plus Δ de celle d'énergie minimale :

$$\mathcal{S}_\Delta = \{S \text{ tel que } E(S) - \text{MFE}(\omega) \leq \Delta\}$$

Une première idée, naturelle dans le contexte de la programmation dynamique, consiste à calculer par programmation dynamique, pour chaque région $[i, j]$, la liste exhaustive des structures Δ-sous optimales réalisables sur la région. Les listes associées aux différentes régions devraient alors être mémorisées dans une matrice spécifique. Cependant, une telle stratégie aurait une complexité mémoire en $\Theta(n^3 \times M)$, où M est le nombre de structures Δ-sous optimales, et deviendrait rapidement prohibitive même pour des ARN de taille modeste.

Le principe de Wuchty *et al*. (1999) consiste à modifier la fonction de remontée, de façon à garantir que chacun des appels récursifs permette la génération d'au moins une structure sous-optimale admissible. Pour cela, on introduit dans la remontée un paramètre Δ, représentant un *budget en sous-optimalité*. Quand on choisit un cas sous-optimal dans la programmation dynamique, on met à jour ce paramètre dans les appels récursifs, afin de tenir compte de l'impossibilité de dériver une structure optimale dans ce cas. On obtient alors l'Algorithme 3, qu'on exécute après un calcul préalable de la matrice de programmation dynamique pour obtenir toutes les structures Δ sous-optimales.

1.2.2.1. *Correction de l'algorithme*

Proposition 3. *Initialisé avec $\Delta \geq 0$ et la région $\sigma := \{[1, n]\}$ l'algorithme 3 renvoie l'ensemble des structures Δ-sous optimales telles que $E(S) \leq m_{1,n} + \Delta$.*

Démonstration. Commençons pas remarquer que, initialement appelé avec $\Delta \geq 0$, `Subopt` préserve cette propriété lors de ses appels récursifs, comme le garantissent les tests des lignes 11, 14, et 18. On suppose donc sans perte de généralité que $\Delta \geq 0$, et on considère une généralisation de la Propriété 3.

Lemme 1. *Etant donné un ARN ω de longueur n, et m la matrice, préalablement calculée, des énergies minimales associées aux régions.*
Alors, pour toute liste σ, toute structure S_p, et toute tolérance $\Delta \geq 0$, la fonction `Subopt`$(\sigma, S_p, \Delta, m, \omega)$ *produit toutes les structures S, étendant S_p avec des structures pour chacune des régions de σ, telles que*

$$E(S) \leq \Delta + \sum_{[i,j]\in\sigma} m_{i,j} + \sum_{(a,b)\in S_p} E^{\omega}_{a,b}. \qquad [1.2]$$

Pour prouver le Lemme 1, on va raisonner sur la *longueur cumulée* $l(\sigma) := \sum_{[i,j]\in\delta} j - i + 1$ des régions de σ. On remarque tout d'abord que, quand $l(\sigma) = 0$,

ALGORITHME 3 : Remontée pour les structures Δ-suboptimales.

Entrée : σ – Ensemble de régions considérées (initialement $\sigma = \{[1, n]\}$)
S_p – Structure secondaire partielle (initialement $S_p = \varnothing$)
Δ – Tolérance (résiduelle), $\Delta \geq 0$
m – Matrice de prog. dyn., préalablement calculée selon l'Équation [1.1]
ω – ARN de taille n
Sortie : $\mathcal{S}_\Delta$ – Structures Δ-sous optimales sur la région $[i, j]$ ordonnées par énergie croissante

Fonction `Subopt` $(\sigma, S_p, \Delta, m, \omega)$**:**
  si $\sigma = \varnothing$ **alors**
    retourner $\{S_p\}$ // La structure partielle S_p est Δ sous-optimale
  sinon
    $[i, j] \leftarrow \texttt{pop}(\sigma)$ // Retire la première région de la pile σ
    si $j - i \leq \theta$ **alors**
      retourner `Subopt`$(\sigma, S_p, \Delta, m, \omega)$ // Traitement des autres régions dans σ
    sinon
      $\mathcal{A} \leftarrow \varnothing; \mathcal{B} \leftarrow \varnothing; \mathcal{C} \leftarrow \{\varnothing\}_{k=i}^{j}$
      ▶ Cas A : Position i laissée sans partenaire
      $\delta_i \leftarrow m_{i+1,j} - m_{i,j}$ // Distance minimale à l'optimale si i libre
      si $\Delta - \delta_i \geq 0$ **alors** // $\exists$ struct. Δ-sous optimale où i est libre
        $\mathcal{A} \leftarrow$ `Subopt`$([i + 1, j] \circ \sigma, S_p, \Delta - \delta_i, m, \omega)$
      ▶ Cas B : Positions i et j forment une paire de base
      $\delta_{i,j} \leftarrow (m_{i+1,j-1} + E^{\omega}_{i,j}) - m_{i,j}$ // Distance min. à l'opt. si (i, j) appariées
      si $\Delta - \delta_{i,j} \geq 0$ **alors** // $\exists$ struct. Δ-sous optimale appariant i et j
        $\mathcal{B} \leftarrow$ `Subopt`$([i + 1, j - 1] \circ \sigma, \{(i, j)\} \cup S_p, \Delta - \delta_{i,j}, m, \omega)$
      ▶ Cas C : Position i appariée à $k < j$
      pour $k \leftarrow i + \theta + 1$ **à** $j - 1$ **faire**
        $\delta_{i,k} \leftarrow (m_{i+1,k-1} + m_{k+1,j} + E^{\omega}_{i,k}) - m_{i,j}$ // Dist. min. si (i, k) appariés
        si $\Delta - \delta_{i,k} \geq 0$ **alors** // $\exists$ struct. Δ-sous optimale appariant i et k
          $\sigma_k \leftarrow [i + 1, k - 1] \circ [k + 1, j] \circ \sigma$
          $\mathcal{C}_k \leftarrow$ `Subopt`$(\sigma_k, \{(i, k)\} \cup S_p, \Delta - \delta_{ik}, m, \omega)$
      retourner $\mathcal{A} \cup \mathcal{B} \cup \bigcup_{k=i}^{j} \mathcal{C}_k$

ie quand $\sigma = \varnothing$, la structure S_p renvoyée par l'algorithme est telle que

$$E(S_p) = \sum_{(a,b)\in S_p} E^{\omega}_{a,b} \leq \sum_{(a,b)\in S_p} E^{\omega}_{a,b} + \Delta$$

et satisfait donc aux conditions de l'Equation [1.2]. Il s'agit en outre de la seule extension possible pour la structure passée en argument, et on conclue donc la validité du Lemme quand $l = 0$, fournissant ainsi une base pour une preuve par récurrence.

On suppose maintenant admise la correction du Lemme 1 pour toute liste de régions σ, de longueur cumulée $l < l^\star$, toute valeur $\Delta \geq 0$ et toute structure S_p. Considérons une liste de régions $\sigma := [i, j] \circ \sigma'$ de longueur cumulée $l^\star$. L'énergie

minimale accessible à partir d'un couple (σ, S_p) est

$$m(\sigma, S_p) := \sum_{[i,j]\in\sigma} m_{i,j} + \sum_{(a,b)\in S_p} E^{\omega}_{a,b}.$$

Choisir un cas de la décomposition revient à se limiter à un sous-ensemble de structures, et l'énergie minimale accessible devient alors $m' \geq m(\sigma, S_p)$. On associe à chacun des cas une *perte d'optimalité*

$$\delta := m' - m(\sigma, S_p) \geq 0$$

à chacun des cas de la décomposition.

Si $\delta > \Delta$, alors pour toute structure S issue d'appels récursifs successifs, on a $E(S) \geq m' = m(\sigma, S_p) + \delta > m(\sigma, S_p) + \Delta$, et S ne doit alors pas être renvoyée par l'algorithme, entrainant la vacuité de $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{C}_k$ quand $\delta > \Delta$ (lignes 11, 14 et 18).

Quand $\delta \leq \Delta$, les structures produites dépendent des trois cas de la décomposition :

Cas A (i libre) : L'énergie minimale d'une structure laissant i libre est donnée par

$$m([i+1, j]\circ\sigma', S_p) = \sum_{[x,y]\in\sigma'} m_{x,y} + \sum_{(a,b)\in S_p} E^{\omega}_{a,b} = m(\sigma, S_p) + m_{i+1,j} - m_{i,j}.$$

On a donc $\delta = m([i+1, j]\circ\sigma', S_p) - m(\sigma, S_p) = m_{i+1,j} - m_{i,j} \geq 0$. Comme $l([i+1, j]\circ\sigma') = l(\sigma) - 1 < l^\star$, l'hypothèse de récurrence s'applique à l'appel récursif sur σ' et S_p. Celui-ci produit donc toutes les structures S étendant S_p sur $[i+1, j]\circ\sigma'$, telles que

$$E(S) \leq m([i+1, j]\circ\sigma', S_p) + \Delta - \delta = m(\sigma, S_p) + \Delta.$$

L'ensemble $\mathcal{A}$ coïncide donc avec la restriction des extensions de S_p sur σ, où i est laissé libre.

Cas B (i apparié à j) : Le cas où i est apparié à j est similaire, mais induit une perte d'optimalité $\delta = E^{\omega}_{i,j} + m_{i+1,j-1} - m_{i,j}$. On a encore $l([i+1, j-1]\circ\sigma') = l(\sigma) - 2 < l^\star$ et l'hypothèse de récurrence implique la correction de l'appel récursif, qui produit donc toutes les structures S, extensions de $S_p \cup \{(i, j)\}$ sur $[i+1, j-1]\circ\sigma'$ telles que

$$E(S) \leq m([i+1, j-1]\circ\sigma', \{(i, j)\} \cup S_p) + \Delta - \delta = m(\sigma, S_p) + \Delta.$$

L'ensemble $\mathcal{B}$ représente donc les extensions de S_p sur σ, satisfaisant [1.2], et appariant i et j.

Cas C (i apparié à $k < j$) Dans le cas d'un appariement de i à $k < j$, on a $\delta = E^{\omega}_{i,k} + m_{i+1,k-1} + m_{k+1,j} - m_{i,j}$. Pour toute valeur de k, l'appel récursif porte sur $\sigma_k := [i+1, k-1] \circ [k+1, j] \circ \sigma$ tel que $l(\sigma_k) = l(\sigma) - 2$. L'hypothèse de récurrence s'applique donc, et on obtient l'ensemble des structures étendant $S_p \cup \{(i,k)\}$ sur σ_k telles que $E(S) \leq m(\sigma_k, \{(i,k)\} \cup S_p) + \Delta - \delta = m(\sigma, S_p) + \Delta$. L'ensemble $\mathcal{C}_k$ représente donc bien les extensions de S_p sur σ, satisfaisant [1.2], et appariant i et k.

Nous concluons en remarquant que toute structure secondaire sur une région $[i,j]$ est engendrée par un des trois cas ci-dessus. Le Lemme est donc correct pour tout σ tel que $l(\sigma) = l^{\star}$, représentant la fin de la récursion, et nous permet de conclure avec la validité du Lemme 1.

Pour finir, on remarque que, quand $\sigma = \{[1,n]\}$ et $S_p = \varnothing$, on obtient les structures telles que

$$E(S) \leq \Delta + \sum_{[i,j] \in \sigma} m_{i,j} + \sum_{(a,b) \in S_p} E^{\omega}_{a,b} = m_{1,n} + \Delta$$

et la correction du Lemme 1 implique donc celle de la Propriété 3. □

1.2.2.2. *Analyse de complexité*

L'explosion combinatoire des structures Δ-optimales produites par l'algorithme, en nombre exponentiel sur Δ et n, interdit une analyse fine de la complexité en fonction uniquement des paramètres d'entrée. On considère donc la complexité en fonction du nombre M de structures produites, et on montre que `Subopt` peut être exécuté en temps $\mathcal{O}(M \times n^2)$.

Concentrons nous tout d'abord sur la structure de l'arbre T des appels récursifs. On remarque que, initialement appelé avec $\Delta \geq 0$, `Subopt` n'effectue que des appels récursifs où $\Delta \geq 0$, ce qu'on peut vérifier aux lignes 11, 14 et 18 du pseudocode. De plus, quand $\sigma \neq \varnothing$ et $\Delta \geq 0$, au moins un de ces tests est satisfait, donc les feuilles de T correspondent au cas $\sigma = \varnothing$, produisant une unique structure $\{S_p\}$ (ligne 3). Ces structures sont distinctes deux à deux (décomposition non-ambiguë), et donc en nombre M puisque toute structure produite est propagée et remonte jusqu'à la racine de T, où elle est renvoyée. Par ailleurs, la hauteur de T est au plus n, car la taille cumulée des régions de σ est strictement décroissante au cours des appels récursifs successifs. Le nombre de noeuds internes (non-racine) est donc au plus $M \times n$ car, à toute profondeur $p \leq n$, le nombre de noeuds à profondeur p est borné par M (sinon il existerait plus de M feuilles dans T).

On obtient la complexité annoncée de $\mathcal{O}(M \times n^2)$ en remarquant que, en excluant les appels récursifs, chaque exécution de `Subopt` requière un nombre au pire linéaire

d'opérations élémentaires. Pour cela, on devra cependant choisir des structures de données adaptées, permettant l'ajout à des listes/piles, et l'union disjointe d'ensembles en temps $\mathcal{O}(1)$. En pratique, des listes suffiront dans les deux cas, permettant une implémentation relativement aisée.

1.2.2.3. *Pour aller plus loin*

Le principe fondamental de l'algorithme, la mise à jour d'une tolérance Δ en fonction des choix effectués lors du backtrack, généralise des travaux antérieurs de Waterman and Byers (1985), et peut être légèrement amélioré grâce à des techniques issues du traitement des langages naturels (Huang and Chiang 2005).

La remontée sous-optimale peut être adaptée à n'importe quel algorithme basé sur un schéma de programmation dynamique non-ambiguë. Elle reste correcte pour une décomposition ambiguë, engendrant certaines structure de plusieurs façons. Cependant, cette multiplicité induit alors un surcoût typiquement exponentiel en n, limitant son utilisation en pratique et motivant la recherche de décompositions non-ambiguës.

1.2.3. *Repliement comparatif : Alignement et repliement conjoint d'ARN*

Une dernière catégorie d'approches pour la prédiction de de repliement tire partie de la conservation de la structure au sein de nombreuses familles fonctionnelles d'ARN non-codants. Si un alignement d'ARN est disponible, alors il suffit de replier simultanément ses ARN de façon à minimiser l'énergie libre cumulée. Cependant, un alignement est parfois difficile à trouver en l'absence de repliement, et on rencontre ainsi un problème de dépendance circulaire (*qui, de l'oeuf et de la poule, apparait est nait le premier ?*).

L'idée de Sankoff (Sankoff 1985), à l'origine de nombreux algorithmes ultérieurs (ainsi que d'une petite centaine de logiciels de méthodes prédictives) consiste à résoudre les problèmes du repliement et de l'alignement simultanément. Comme tout problème d'alignement multiple, le problème ainsi décrit est en général NP-complet (Wang and Jiang 1994). Cependant la restriction du problème à deux séquences d'ARN est déjà pertinente, dû à la présence de *mutations compensatoires*, des paires de mutations maintenant la possibilité d'un appariement, au cours de l'évolution des ARN. Dans ce cadre d'alignement/repliement d'une paire d'ARN, le problème admet une solution exacte en $\Theta(n^6)$, basé sur un produit de deux schéma de programmation dynamique décrit visuellement à la Figure 1.5.

1.2.3.1. *Modèle d'alignement/repliement conjoint*

Décrivons plus précisément la notion d'*alignement/repliement conjoint* pour une paire d'ARN (u, v). On rappelle tout d'abord qu'un *alignement* de deux séquences d'ARN peut être défini comme une paire de chaines de caractères $A = (u', v')$ sur un alphabet étendu $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}, \mathtt{-}\}$, où le caractère - représente une insertion/déletion (indel) telles que :

– Les deux séquences sont de même longueur $|u'| = |v'| \geq \max(|u|, |v|)$;

– u (resp. v) est obtenu à partir de u' (resp. v') en supprimant les indels (-).

Par exemple, les séquences d'ARN $u :=$ `ACGU` et $v :=$ `AGAU` admettent (entre autres) les alignements suivants :

$$A_1 := \begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & & & 4 \end{array} \\ u' \to & \begin{array}{cccccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{-} & \mathtt{-} & \mathtt{U} \end{array} \\ v' \to & \begin{array}{cccccc} \mathtt{A} & \mathtt{-} & \mathtt{G} & \mathtt{A} & \mathtt{U} & \end{array} \\ & \begin{array}{cccccc} 1 & & 2 & 3 & 4 & \end{array} \end{array} \qquad A_2 := \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \hline \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{U} \\ \mathtt{A} & \mathtt{G} & \mathtt{A} & \mathtt{U} \\ \hline 1 & 2 & 3 & 4 \end{array} \qquad A_3 := \begin{array}{cccccccc} 1 & 2 & 3 & 4 & & & & \\ \hline \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{U} & \mathtt{-} & \mathtt{-} & \mathtt{-} & \mathtt{-} \\ \mathtt{-} & \mathtt{-} & \mathtt{-} & \mathtt{-} & \mathtt{A} & \mathtt{G} & \mathtt{A} & \mathtt{U} \\ \hline & & & & 1 & 2 & 3 & 4 \end{array}$$

Chacun des alignements induit un ensembles de *correspondances*, appelés *(mis)matches*, entre certaines des positions des deux ARN. Par exemple, l'alignement A_1 ci-dessus induit l'ensemble de matches $\{(1,1),(3,2),(4,4)\}$, l'alignement A_2 induit les matches $\{(1,1),(2,2),(3,3),(4,4)\}$ et A_3 ne présente aucun match ($\varnothing$).

Tous les alignements ne sont pas également réalistes, et des modèles d'évolution permettent ainsi d'associer à tout alignement une probabilité. Dans un vision maximisant la parcimonie, ceux-ci sont en général définis comme un produit de probabilités indépendantes, associées à d'évènements évolutifs suggérés par l'alignement. Par exemple, le match $(1,1)$ dans A_1 suggère la présence du nucléotide `A` dans une séquence ancestrale, alors que le `C` en deuxième colonne pourrait avoir été acquis récemment par u (ou perdu par v). Enfin, la mise en correspondance de deux nucléotides distincts, par exemple dans la deuxième colonne de A_2, suggère une mutation suite à la spéciation/duplication de l'ARN considéré.

En estimant les probabilités de ces évènements, on arrive donc à une définition pour la probabilité d'un alignement, obtenue en multipliant les probabilités des évènements représentés par les colonnes de l'alignement :

$$\mathbb{P}(A \mid u, v) = \prod_{\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right] \in A} \mathbf{P}\mu_{x,y} \prod_{\left[\begin{smallmatrix} x \\ - \end{smallmatrix}\right] \in A} \mathbf{P}\iota_x \prod_{\left[\begin{smallmatrix} - \\ y \end{smallmatrix}\right] \in A} \mathbf{P}\delta_y$$

où $\mathbf{P}\mu_{x,y}$ représente la probabilité d'une conservation ($x = y$) ou substitution ($x \neq y$), et $\mathbf{P}\iota_x$ (resp. $\mathbf{P}\delta_y$) celle d'une insertion dans u (resp. dans v).

Un alignement/repliement conjoint (A, S) consiste alors à superposer une structure secondaire S sur un alignement $A = (u', v')$, la structure impliquant alors les colonnes de l'alignement. Pour toute paire de base (i, j) dans S, au moins une des conditions suivantes est satisfaite :

– La paire (u'_i, u'_j) contient des nucléotides (indels interdits) appariables ;

– La paire (v'_i, v'_j) contient des nucléotides (indels interdits) appariables.

Ces deux conditions peuvent aussi être simultanément respectées, et ainsi permettre de tenir compte d'une possible co-évolution des deux positions. Au contraire, si une seule des séquence permet l'appariement, alors sa présence dans une structure commune en devient moins probable. On définit alors les structures $S \rightsquigarrow (S, T)$, S et T induites par S sur u et v, obtenues en retirant les indels (-) et les paires de bases impliquant au moins un indel.

Dans une première approche, la probabilité d'un alignement A, en conjonction avec une structure commune S pour ses deux séquences, peut alors être (arbitrairement) définie comme

$$\mathbb{P}(A, S \mid u, v) \propto \mathbb{P}(A \mid u, v)\, \mathbb{P}(S \mid u)\, \mathbb{P}(T \mid v) \prod_{(a,b)\in S} \mathbf{P}\pi^{u'_a, u'_b}_{v'_a, v'_b} \qquad [1.3]$$

où $\mathbb{P}(S^\star \mid \omega)$ est la probabilité de Boltzmann de $S^\star$ pour une séquence ω, et $\mathbf{P}\pi^{x_1, y_1}_{x_2, y_2}$ est la probabilité d'une *substitution de paire de base*, impliquant des nucléotides (x_1, y_1) dans u et (x_2, y_2) dans v.

En particulier, $\mathbf{P}\pi$ permet de valoriser les *mutations compensatoires* :

Paire de colonnes $\boxed{\begin{matrix} x \\ y \end{matrix}}$ et $\boxed{\begin{matrix} a \\ b \end{matrix}}$ telle que $x \neq a, y \neq b$ et (x, a) et (y, b) appariables .

De telles mutations sont souvent interprétées comme indiquent une pression de sélection positive sur la structure, et sont utilisées en modélisation comparative des ARN depuis les travaux fondateurs de Michel and Westhof (1990).

On cherche alors à maximiser cette probabilité, ce qui revient à maximiser la partie droite de l'équation [1.3]. En pratique, pour éviter d'éventuels problèmes numériques, on considère donc plutôt une version logarithmique, équivalente du point de vue de l'optimisation. Les fonctions de partitions contribuent alors un facteur constant, indépendant de la structure ou de l'alignement, et peuvent donc être négligées pour l'optimisation. La fonction objectif devient alors

$$\begin{aligned} F(A, S) = &\sum_{\boxed{\begin{matrix} x \\ y \end{matrix}} \in A} \mu_{x,y} + \sum_{\boxed{\begin{matrix} x \\ - \end{matrix}} \in A} \iota_x + \sum_{\boxed{\begin{matrix} - \\ y \end{matrix}} \in A} \delta_y \\ &- (E(S, u) + E(T, v)) + \sum_{(a,b)\in S} \pi^{u'_a, u'_b}_{v'_a, v'_b} \end{aligned} \qquad [1.4]$$

où μ, ι, δ et π représentent les logarithmes naturels, multipliés par kT, respectifs de $\mathbf{P}\mu$, $\mathbf{P}\iota$, $\mathbf{P}\delta$ et $\mathbf{P}\pi$.

Alignement/repliement conjoint

Entrée: $u, v \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}^\star$; Matrices ι, δ, μ, et π.
Sortie: Couple alignement/structure $(A^\star, S^\star)$, le plus probable selon [1.4] :

$$F(A^\star, S^\star) = \max_{A,S} F(A, S)$$

1.2.3.2. *Algorithme et complexité*

Le problème ci-dessus peut aussi être résolu par un algorithme de programmation dynamique polynomial. Il repose sur une simulation de tous les alignements possibles au cours du repliement, illustré par la Figure 1.5. On en déduit immédiatement une équation de programmation dynamique pour calculer le *score logarithmique* maximal sur des régions $[i, j]$ et $[k, l]$ de u et v respectivement.

Pour des régions vides sur u et/ou v, on a

$$f^{i,j<i}_{k,l<l} := 0 \qquad \text{// Régions de } u \text{ et } v \text{ toutes les deux vides}$$

$$f^{i,j<i}_{k,l\geq k} := -m^v_{k,l} + \sum_{c\in v_{k,l}} \delta_c \qquad \text{// Région de } v \text{ vide} \rightarrow \text{On replie et insère la région de } u$$

$$f^{i,j\geq i}_{k,l<k} := -m^u_{i,j} + \sum_{c\in u_{i,j}} \iota_c \qquad \text{// Région de } u \text{ vide} \rightarrow \text{On replie et supprime la région de } v$$

où $m^\omega_{i,j}$ représente l'énergie minimale d'un repliement de ω sur la région $[i, j]$, calculé comme vu en section 1.2.1. Outre les énergies induites par les repliements indépendants, les scores optimaux doivent tenir compte de l'insertion/suppression complète de la région non-vide, d'où les sommes ci-dessus.

Dans le cas général, on considère des régions non-vides dans u et v, et on a alors

$$f^{i,j\geq i}_{k,l\geq k} := \max \begin{cases} \text{► Cas A : Positions } i \text{ et/ou } k \text{ non apparié(es)} \\ \max\limits_{b_i,b_k\in\{0,1\}^2} b_i\,\overline{b_k}\,\iota_{u_i} + \overline{b_i}\,b_k\,\delta_{v_k} + b_i\,b_k\,\mu_{u_i,v_k} + f^{i+b_i,j}_{k+b_k,l} \\ \text{► Cas B/C : Appariements } (i,x) \text{ et/ou } (k,y) \\ \max\limits_{\substack{\binom{x}{y}=\binom{i+\theta+1}{k+\theta+1};\\ \binom{b_i,b_x}{b_k,b_y}\in[0,1]^4 \\ b_ib_x+b_kb_y\geq 1}} \left| \begin{array}{l} b_i\,\overline{b_k}\,\iota_{u_i} + \overline{b_i}\,b_k\,\delta_{v_k} + b_x\,\overline{b_y}\,\iota_{u_x} + \overline{b_x}\,b_y\,\delta_{v_y} \\ \quad + b_i\,b_k\,\mu_{u_i,v_k} + b_x\,b_y\,\mu_{u_x,v_y} \\ \quad + b_i\,b_x\,b_k\,b_y\,\pi^{u_i,u_x}_{v_k,v_y} \\ \quad + b_i\,b_x\,E^u_{i,x} + b_k\,b_y\,E^v_{k,y} \\ \quad + f^{i+b_i,x-b_x}_{k+b_k,y-b_y} + f^{x+1,j}_{y+1,l} \end{array} \right. \end{cases} \qquad [1.5]$$

où $\overline{b_p} := (1 - b_p)$ pour toute position p.

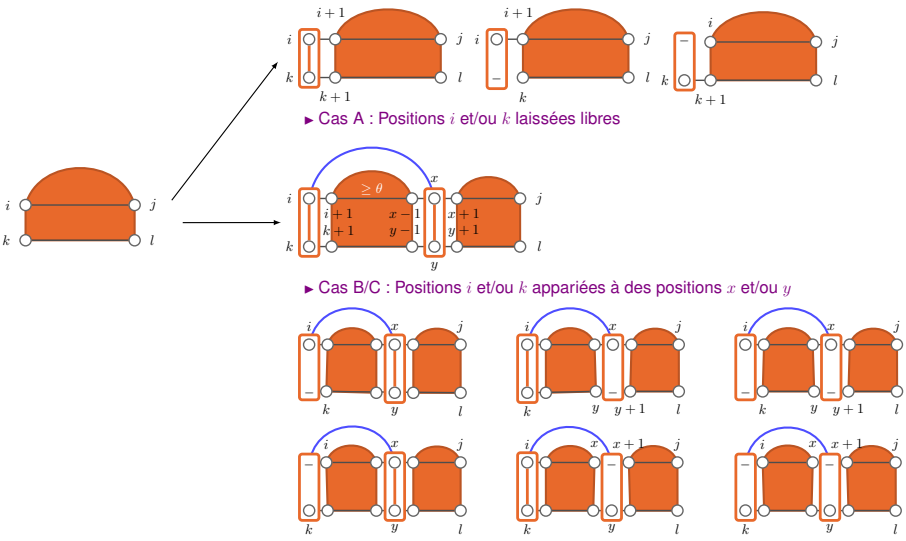


Figure 1.5. *Décomposition de l'espace des alignements/repliements conjoints à la base de l'algorithme de Sankoff. Pour deux séquences d'ARN restreintes aux régions $[i, j]$ et $[k, l]$, on distingue les cas libres et appariés, et tous les alignements locaux.*

Chacun des b_p exprime l'implication ($b_p = 1$) ou non ($b_p = 0$) d'une position p dans une colonne de l'alignement engendré. Ces variables booléennes permettent de factoriser les (très) nombreux cas dans la décomposition, induits par l'énumération des alignements, en n'incorporant que des termes pertinents dans chacun des cas.

Par exemple, considérons le terme $b_i\, b_k\, \mu_{u_i,v_k}$. Si les positions i et k sont tous deux alignées ($b_i = b_k = 1$), on a alors $b_i\, b_k\, \mu_{u_i,v_k} = \mu_{u_i,v_k}$, correspondant au terme de conservation/substitution attendu pour un match de i et k. En revanche, si une des deux positions n'est pas alignée ($b_i = 0$ ou $b_k = 0$), on a alors $b_i\, b_k\, \mu_{u_i,v_k} = 0$, et le score ne contient pas de contribution du match. De la même façon, on a :

$b_p\, \overline{b_q}\, \iota_{u_p} \rightarrow$ Score d'insertion seulement si p est sans partenaire ($\overline{q}$) ;

$\overline{b_p}\, b_q\, \delta_{u_q} \rightarrow$ Score de suppression seulement si q est sans partenaire ($\overline{p}$) ;

$b_p\, b_q\, b_r\, b_s\, \pi^{u_p,u_q}_{v_r,v_s} \rightarrow$ Score de substitution de paire des bases seulement si les quatre positions impliquées sont deux à deux alignées ;

$b_p\, b_q E^{\omega}_{p,q} \rightarrow$ Energie de la paire de paire seulement si la paire est peuplée.

L'algorithme de Sankoff calcule alors les termes de cette récurrence par programmation dynamique en procédant, dans sa boucle extérieure, par ordre croissant sur la *taille cumulée* $N(i,j,k,l)$ des régions considérées, définie telle que :

$$N(i,j,k,l) = |[i,j]| + |[k,l]| = j - i + k - l + 2$$

et dans n'importe quel ordre pour les autres indices/boucles. Une remontée vient alors compléter l'algorithme, et reconstruire le repliement/alignement optimal.

L'exécution de cet algorithme requière un temps $\Theta(n^6)$ pour deux séquences u et v ayant même longueur n, et un espace $\Theta(n^4)$ en mémoire. Plus précisément, la complexité de cet algorithme est en $\Theta(|u|^3 \cdot |v|^3)$ en temps, et $\Theta(|u|^2 \cdot |v|^2)$ en mémoire. La décomposition sous-jacente à l'algorithme de Sankoff peut aussi être généralisée à M séquences, mais sa complexité augmente alors jusqu'à $\Theta(n^{3M})$ en temps et $\Theta(n^{2M})$ en mémoire.

1.2.3.3. *Pour aller plus loin*

Le principe de l'algorithme de Sankoff est au coeur de virtuellement toutes les approches pour la prédiction comparative. La qualité de ses prédictions sont nettement supérieures à celles obtenues par minimisation d'énergie et apprentissage machine. Cependant, sa complexité élevée, en particulier en mémoire, empêche son utilisation directe pour des ARN de longueur supérieure à ~ 100 nucleotides. C'est pourquoi on trouve, dans les implémentations modernes, diverses astuces de calcul et heuristiques qui permettent un passage à l'échelle sans sacrifice substantiel de capacité prédictive (voir `SPARSE` (Will *et al*. 2015), de complexité $\Theta(n^2)$).

Le modèle d'alignement peut être étendu dans plusieures directions :

– Tout d'abord, il peut tenir compte du modèle d'énergie de Turner (Turner and Mathews 2010), mais aussi de modèles évolutifs plus complexes, par exemple en associant un arbre phylogénétique à la prédiction pour tenir compte de la distance d'évolution au moment d'interpréter une mutation compensatoire.

– Le coût associé à une séquence de g indels consécutif peut être défini comme une fonction affine $\alpha \times g + \beta$, au lieu de $\alpha' \times g$ dans notre modèle. Il être optimisé par une variante de l'algorithme de Sankoff, sans surcoût algorithmique grâce à une astuce générique due à Gotoh (1982). Une astuce similaire peut aussi être utilisée pour lever l'ambiguïté induite par l'alignement (essentiellement due à la commutativité des indels), ouvrant la porte à un alignement sous l'hypothèse d'une préservation de l'ensemble de Boltzmann (Will *et al*. 2012).

1.3. Approches ensemblistes

1.3.1. *Calcul de la fonction de partition*

Même si les solutions sous-optimales permettent la considération des structures dans une certaine région de stabilité, elles ne permettent pas d'apprécier la

représentativité statistique des ensembles de structures produites. En particulier, les structures sous-optimales, produites par l'algorithme pour une tolérance Δ donnée, peuvent être extrêmement similaires sans être pour autant représentatives de l'ensemble de Boltzmann. De la même façon, à l'équilibre thermodynamique, la structure d'énergie minimale peut être associée à une probabilité, certes maximale par définition, mais dérisoire.

Nous pouvons formaliser cette notion de représentativité à l'aide de concepts issus de la mécanique statistique. Pour cela, on rappelle qu'à l'équilibre thermodynamique, l'ensemble des structures $S \in \mathcal{S}_\omega$ possibles pour un ARN ω, sont observées selon une *distribution de Boltzmann* :

$$\mathbb{P}(S \mid \omega) = \frac{e^{-\frac{E(S)}{kT}}}{\mathcal{Z}} \qquad [1.6]$$

où T est la température (K), k est la constante de Boltzmann ($1.987.10^{-3}$ kcal.mol^{-1}.K^{-1}), et $\mathcal{Z}$ est la *fonction de partition*, définie comme

$$\mathcal{Z} = \sum_{S \in \mathcal{S}} e^{-E(S)/kT}. \qquad [1.7]$$

La fonction de partition est essentielle à l'adoption d'un point de vue statistique sur les ensemble de conformations. Par exemple, la probabilité de la structure d'énergie minimale, et donc une évaluation partielle de sa représentativité, est donnée par $e^{-m_{1,n}/kT}/\mathcal{Z}$. Plus généralement, le calcul de la fonction de partition est un prérequis indispensable à des approches par échantillonnage, présentées en Section 1.3.2, ou le calcul exact des propriétés moyennes de l'ensemble, décrit en Section 1.3.3.

En pratique, nous avons non seulement besoin de $\mathcal{Z}$, mais aussi de $\mathcal{Z}_{i,j}$, la fonction de partition restreinte à l'ensemble $\mathcal{S}_{i,j}$ des structures adoptées sur une région $[i, j]$. On note que ces fonctions de partitions permettent de retrouver la fonction de partition globale du système à travers $\mathcal{Z} := \mathcal{Z}_{1,n}$. Le calcul de terme pour toutes les régions $[i, j]$ constitue donc un problème, potentiellement complexe, de comptage pondéré.

Calcul de la fonction de partition $\mathcal{Z}$

Entrée: Séquence $\omega \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}^+$, $|\omega| = n$.
Sortie: Fonction de partition $\mathcal{Z}_{i,j}$ pour toute région $[i, j]$, définie telle que

$$\mathcal{Z}_{i,j} = \sum_{S \in \mathcal{S}_{i,j}} e^{-E(S)/kT}$$

ALGORITHME 4 : Calcul de la fonction de partition $\mathcal{Z}$

Entrée : ω – ARN de taille n
Sortie : $\mathcal{Z}$ – Fonction de partition Z

```
1  Fonction FonctionPartition (ω):
2      Z ← MatriceVide(n × n)
       // Initialiser à 1 toutes les valeurs jusqu'à θ de la diagonale.
3      pour i ← 1 à n faire
4          pour j ← i à min (i + θ, n) faire
5              Z_{i,j} ← 1
6      pour i ← n à 1 faire
7          pour j ← i + θ + 1 à n faire
               ▶ Cas A : Position i laissée sans partenaire
8              Z_{i,j} ← Z_{i+1,j}
               ▶ Cas B : Positions i et j forment une paire de base
9              Z_{i,j} ← Z_{i,j} + Z_{i+1,j-1} × e^{-E^ω_{i,j}/kT}
               ▶ Cas C : Position i appariée à k < j
10             pour k ← i + θ + 1 à j − 1 faire
11                 Z_{i,j} ← Z_{i,j} + Z_{i+1,k-1} × Z_{k+1,j} × e^{-E^ω_{i,k}/kT}
12     retourner Z
```

L'algorithme 4 permet de calculer la fonction de partition $\mathcal{Z}$. Alors que la somme est sur un nombre exponentiel d'élément, la taille de $\mathcal{S}$, nous allons montrer que nous pouvons en fait calculer $\mathcal{Z}$ très efficacement, en temps polynomial sur la taille de l'ARN considéré.

En fait, nous avons déjà produit l'algorithme, à un changement d'algèbre prêt ! En effet, notre implémentation de `DescenteMin` dans l'algorithme 1 peut être légèrement modifiée de façon à calculer $\mathcal{Z}_{i,j}$ au lieu de $m_{i,j}$. Pour cela, il suffit de remplacer les sommes par des produits, les minimisations par des sommes, et d'élever les termes énergétiques constants à leur facteur de Boltzmann :

$$\min \to + \qquad + \to \times \qquad E \to e^{-E/kT}$$

Nous obtenons alors l'Algorithme 4, qui permet de calculer la fonction de partition $\mathcal{Z}$.

1.3.1.1. *Correction des algorithmes*

On commence par une remarque purement technique, sur laquelle va reposer assez largement notre preuve par induction. Précisément, soit $E(S) = E_1 + \cdots + E_l$ l'énergie d'une structure S, décomposable en l termes, alors on a

$$\prod_{i=1}^{l} e^{-E_i/kT} = e^{-\sum_{i=1}^{l} E_i/kT} = e^{-E(S)/kT} \qquad [1.8]$$

Armé de cette propriété, nous pouvons maintenant établir la correction du remplissage de la matrice.

Proposition 4. *L'algorithme 4 calcule correctement la fonction de partition de ω pour chaque région.*

Démonstration. Nous allons procéder par *induction sur la longueur des régions* $[i, j]$, et prouver que $\mathcal{Z}_{i,j}$ contient la fonction de partition restreinte à la sous-séquence $\omega_{i,j}$. Dans le *cas de base*, lorsque $i - j \leq \theta$, il n'y a qu'une structure possible, sans paire de base et donc d'énergie nulle, son facteur de Boltzmann est donc $e^{-0/kT} = 1$. Or, l'initialisation affecte à $\mathcal{Z}_{i,j}$ la valeur 1 pour toute région de longueur au plus $\theta + 1$, et nous obtenons donc le résultat escompté.

Supposons maintenant la validité de la proposition pour toute région $[i', j']$ de longueur $j' - i' + 1 < n^\star$, c'est à dire que la valeur $\mathcal{Z}_{i',j'}$ coincide bien avec la fonction de partition restreinte à $\omega_{i',j'}$. Considérons maintenant un région $[i, j]$ de longueur $n^\star$, lorsque nous calculons $\mathcal{Z}_{i,j}$ nous avons alors trois cas possibles :

Cas A : Position i laissée sans partenaire. Comme $j-i+1 < n^\star$, l'hypothèse d'induction s'applique et, en conjonction avec l'absence de contribution énergétique des positions non-appariées, implique que :

$$\mathcal{Z}_{i+1,j} = \sum_{S' \in \mathcal{S}_{i+1,j}} e^{\frac{-E(S')}{kT}} = \sum_{S' \in \mathcal{S}_{i+1,j}} e^{\frac{-E(\bullet S')}{kT}} = \sum_{S \in \bullet\mathcal{S}_{i+1,j}} e^{\frac{-E(S)}{kT}}.$$

En d'autre termes, $\mathcal{Z}_{i+1,j}$ coincide avec la fonction de partition sur $[i, j]$, restreinte aux structures laissant i libre, qu'on notera $\mathcal{Z}_{\bullet S}$ par la suite.

Cas B : Positions i appariée à j. L'hypothèse d'induction s'applique à $[i+1, j-1]$ et, en remarquant que la paire (i, j) contribue une énergie $E^\omega_{i,j}$. On a donc :

$$e^{\frac{-E^\omega_{i,j}}{kT}} \times \mathcal{Z}_{i+1,j-1} = e^{\frac{-E^\omega_{i,j}}{kT}} \sum_{S' \in \mathcal{S}_{i+1,j-1}} e^{\frac{-E(S')}{kT}} = \sum_{S' \in \mathcal{S}_{i+1,j-1}} e^{\frac{-(E^\omega_{i,j}+E(S'))}{kT}}$$

$$= \sum_{S' \in \mathcal{S}_{i+1,j-1}} e^{\frac{-E((S'))}{kT}} = \sum_{S \in (\mathcal{S}_{i+1,j-1})} e^{\frac{-E(S)}{kT}}$$

Le calcul proposé dans l'algorithme recouvre donc toutes les structures appariant i à j, dont on dénote par $\mathcal{Z}_{(S)}$ la fonction de partition.

Cas C : Position i appariée à $k < j$. L'algorithme rajoute dans ce cas à la fonction de partition une quantité $\sum_{i+\theta+1}^{j-1} e^{-E^\omega_{i,j}/kT} \mathcal{Z}_{i+1,k-1}\mathcal{Z}_{k+1,j}$, dont les termes

peuvent être supposés corrects d'après l'hypothèse d'induction. On obtient donc :

$$e^{\frac{-E^{\omega}_{i,k}}{kT}}\mathcal{Z}_{i+1,k-1}\mathcal{Z}_{k+1,j} = e^{\frac{-E^{\omega}_{i,k}}{kT}}\sum_{S_1\in\mathcal{S}_{i+1,k-1}} e^{\frac{-E(S_1)}{kT}}\sum_{S_2\in\mathcal{S}_{k+1,j}} e^{\frac{-E(S_2)}{kT}}$$

$$= \sum_{S_1\in\mathcal{S}_{i+1,k-1}}\sum_{S_2\in\mathcal{S}_{k+1,j}} e^{-\frac{E^{\omega}_{i,k}+E(S_1)+E(S_2)}{kT}} = \sum_{S\in(S_{i+1,k-1})\mathcal{S}_{k+1,j}} e^{\frac{-E(S)}{kT}}.$$

Le terme proposé par l'algorithme représente donc la fonction de partition restreinte aux les structures appariant i à k, pour une valeur donnée de k. En sommant sur toutes les valeurs de k sur $[i+\theta+1, j-1]$ on obtient la fonction de partition $\mathcal{Z}_{(S_1)S_2}$ de toutes les structures appariant i à toute autre position que j.

La non-ambiguité de la décomposition, ainsi que le bon sens populaire, implique que les ensembles de structures laissant i libre, appariant i à j, ou appariant i à tout $k < j$, sont disjoints. Par ailleurs, toute structure sur une région $[i, j]$ rentre dans une de ces trois catégories. On en conclut donc que :

$$\mathcal{Z}_{\bullet S} + \mathcal{Z}_{(S)} + \mathcal{Z}_{(S_1)S_2} = \sum_{\substack{S\in\bullet\mathcal{S}_{i+1,j}\bigcup(\mathcal{S}_{i+1,j-1})\\ \bigcup(S_{i+1,k-1})\mathcal{S}_{k+1,j}}} e^{-\frac{E(S)}{kT}} = \sum_{S\in\mathcal{S}_{i,j}} e^{\frac{-E(S)}{kT}} \qquad [1.9]$$

et la validité de la fonction de partition calculée pour toute région de longueur $n < n^{\star}$ implique donc celle de toute région de taille $n^{\star}$. □

Les différences entre les algorithmes 1 et 4 induisent des surcoûts en temps/mémoire indépendants de n : Les termes énergétiques élémentaires sont à $e^{-E/kT}$ au lieu de E, les sommes et produits viennent remplacer les minimums et sommes, et sont toute réalisables en temps supposé constant. Nous conservons donc la complexité de $\Theta(n^3)$.

1.3.1.2. *Pour aller plus loin*

Le changement d'algèbre $(\min, +, E) \to (+, \times, e^{-E/kT})$ peut, en principe, être adapté à n'importe quel problème combinatoire admettant un algorithme par programmation dynamique. Cependant, pour que l'algorithme modifié calcule bien la fonction de partition sur l'espace des solutions, il faudra s'assurer que la décomposition sur laquelle repose l'algorithme soit complète – produise tous les éléments de l'espace de recherche – et non-ambiguë – produise chaque élément au plus une fois.

Le nombre de structures secondaires compatibles avec un ARN ω peut être obtenu aisément à travers un calcul de fonction de partition. Pour cela, il suffit d'affecter à la température T une très grande valeur, et on obtient alors

$$\mathcal{Z} := \mathcal{Z}_{1,n} = \sum_{s\in\mathcal{S}} e^{-E(S)/kT} \underset{T\to+\infty}{\longrightarrow} \sum_{s\in\mathcal{S}} e^0 = \sum_{s\in\mathcal{S}} 1 = |\mathcal{S}|$$

1.3.2. *Echantillonnage statistique*

En calculant la fonction de partition, nous disposons maintenant d'un outil essentiel pour envisager une analyse statistique de l'ensemble des repliements. Cependant, la fonction de partition ne donne essentiellement accès qu'aux probabilités individuelles des structures. Or, celles-ci sont en nombre exponentiel, et leur probabilité est typiquement exponentiellement décroissante sur la longueur n des ANR considérés.

Pour contourner ces difficultés, et avoir accès à des statistiques décrivant l'ensemble de Boltzmann, Ding and Lawrence (2003) introduisent un *algorithme de génération aléatoire*, aussi appelé échantillonage statistique dans la littérature. L'idée consiste à introduire un aspect stochastique dans l'étape de remontée, de façon à produire des structures aléatoires, chacune engendrée selon la *distribution de Boltzmann*

$$\mathbb{P}(S \mid \omega) = \frac{e^{-E(S)/kT}}{\mathcal{Z}}$$

où $\mathcal{Z} = e^{-E(S)/kT}$ est la fonction de partition, calculée au chapitre précédent.

Echantillonnage de structures

Entrée: Séquence $\omega \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}^+$, $|\omega| = n$.
Sortie: Structure S avec probabilité

$$\mathbb{P}(S) = \frac{p_S}{\mathcal{Z}} = \frac{e^{-E(S)/kT}}{\mathcal{Z}}$$

La *remontée stochastique*, décrite dans l'Algorithme 2, permet de résoudre le problème ci-dessous dans le cadre d'un modèle d'énergie simple.

Son principe, illustré en Figure 1.6, repose sur le choix aléatoire, à chaque étape, d'un des cas de la décomposition, avec probabilité proportionnelle à sa contribution à la fonction de partition. En effet, considérons une région $[i, j]$ donnant accès à un ensemble $\mathcal{S}_{i,j}$ de structures et imaginons que chaque cas dans la décomposition donne accès à un sous-ensemble $\mathcal{S}' \subset \mathcal{S}_{i,j}$, associé à une fonction de partition $\mathcal{Z}_{\mathcal{S}'} := \sum_{S \in \mathcal{S}'} e^{-E(S)/kT}$. On propose donc de choisir ce cas avec probabilité :

$$p_{\mathcal{S}'} = \frac{\mathcal{Z}_{\mathcal{S}'}}{\mathcal{Z}_{i,j}}$$

En effet, si l'on dispose d'un générateur de Boltzmann valide pour $\mathcal{S}'$, alors la probabilité d'engendrer un élément de $S' \in \mathcal{S}'$ à partir de $[i, j]$ devient

$$\mathbb{P}(S' \mid [i,j]) = p_{\mathcal{S}'} \times \mathbb{P}(S' \mid \mathcal{S}') = \frac{\mathcal{Z}_{\mathcal{S}'}}{\mathcal{Z}_{i,j}} \times \frac{e^{-E(S')/kT}}{\mathcal{Z}_{\mathcal{S}'}} = \frac{e^{-E(S')/kT}}{\mathcal{Z}_{i,j}}$$

où l'on reconnaît la probabilité visée.

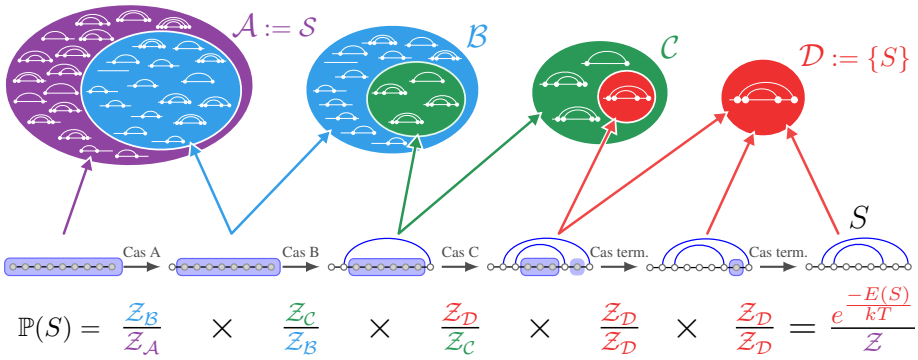


Figure 1.6. *Principe de génération aléatoire par la méthode dite récursive. A chaque étape, on choisit un cas de la décomposition avec prob. proportionnelle à sa contribution dans la fonction de partition. La prob. d'engendrer une structure S est alors égale au produit des prob. des choix opérés, et les numérateur/dénominateurs consécutifs s'annulent deux à deux pour ne laisser que la prob. de Boltzmann de S.*

1.3.2.1. *Correction de l'algorithme*

Proposition 5. *Soit $\mathcal{Z}_{i,j}$ la fonction de partition d'un ARN ω, restreint à sa région $[i,j]$. L'appel* `StructAléatoire`$(i,j,\omega,\mathcal{Z})$ *retourne $S^\star$ avec probabilité :*

$$\mathbb{P}(S) = \frac{e^{-E(S)/kT}}{\mathcal{Z}_{i,j}}$$

Démonstration. Nous allons procéder par *induction sur la longueur des régions* $[i,j]$. Dans le *cas de base*, lorsque $i - j \leq \theta$, il n'y a qu'une structure $S^\star$ possible, la structure vide. Elle est retournée avec probabilité 1 comme décrit à la ligne 2.

Supposons maintenant la propriété vraie pour toute région $[i,j]$ tel que $j-i+1 \leq n-1$. Comme nous l'avons vu pour le calcul de la fonction de partition, nous avons :

$$\overbrace{\mathcal{Z}_{i,j}}^{S} = \overbrace{\mathcal{Z}_{i+1,j}}^{\bullet\mathcal{S}_{i+1,j}} + \overbrace{E^\omega_{i,j} \times \mathcal{Z}_{i+1,j-1}}^{(\mathcal{S}_{i+1,j-1})} + \sum_{k=i+\theta+1}^{j-1} \overbrace{E^\omega_{i,k} \times \mathcal{Z}_{i+1,k-1} \times \mathcal{Z}_{k+1,j}}^{(\mathcal{S}_{i+1,k-1})\mathcal{S}_{k+1,j}}. \quad [1.10]$$

Chacun des termes de la somme représente un sous-ensemble de structures possibles pour $S^\star$, associé à un des cas de la décomposition. Le nombre aléatoire a généré par l'algorithme 5 va servir à identifier le cas de la décomposition, qui sera choisi avec probabilité proportionnelle à sa contribution à la fonction de partition.

Considérons une structure $S^\star$ engendrée par l'algorithme sur une région $j-i+1 \leq n$. On a maintenant trois cas possibles :

ALGORITHME 5 : Engendre une structure S avec probabilité $e^{-E(S)/kT}/\mathcal{Z}$

Entrée : $[i, j]$ – Région considérée
$\mathcal{Z}$ – Fonction de partition, préalablement calculée selon l'Algorithme 4
ω – ARN de taille n
Sortie : S – Structure aléatoire de taille n, engendrée selon la distribution de Boltzmann

1 **Fonction** `StructAléatoire`$(i, j, \mathcal{Z}, \omega)$**:**
2 **si** $j - i \leq \theta$ **alors retourner** $\overbrace{\bullet \ldots \bullet}^{j-i+1}$ // Structure vide est unique → Probabilité 1
3 **sinon**
4 $a \leftarrow$aléatoire$(0, \mathcal{Z}_{i,j})$ // Tirage aléatoire uniforme sur $[0, \mathcal{Z}_{i,j}[$
▶ Cas A : Position i laissée sans partenaire
5 $a \leftarrow a - \mathcal{Z}_{i+1,j}$ // Retrait de la fonc. part. des struct. laissant i libre
6 **si** $a < 0$ **alors** // Vrai quand $a \in [0, \mathcal{Z}_{i+1,j}[$ → Probabilité $\mathcal{Z}_{i+1,j}/\mathcal{Z}_{i,j}$
7 $S_i^\star \leftarrow$ `StructAléatoire`$(i + 1, j, \mathcal{Z}, \omega)$
8 **retourner** $\bullet\, S_i^\star$
▶ Cas B : Positions i et j forment une paire de base
9 $a \leftarrow a - \mathcal{Z}_{i+1,j-1} \times e^{-E^\omega_{i,j}/kT}$ // Retrait de la fonc. part. des struct. appariant i à j
10 **si** $a < 0$ **alors** // Vrai avec prob. $(\mathcal{Z}_{i+1,j} \times e^{-E^\omega_{i,j}/kT})/\mathcal{Z}_{i,j}$
11 $S^\star_{i,j} \leftarrow$ `StructAléatoire`$(i + 1, j - 1, \mathcal{Z}, \omega)$
12 **retourner (** $S^\star_{i,j}$ **)**
▶ Cas C : Position i appariée à $k < j$
13 **pour** $k \leftarrow i + \theta + 1$ **à** $j - 1$ **faire**
14 $a \leftarrow a - \mathcal{Z}_{i+1,k-1} \times \mathcal{Z}_{k+1,j} \times E^\omega_{i,k}$ // Retrait de la fonc. part. appariant i à $k < j$
15 **si** $a < 0$ **alors** // Vrai avec prob. $(\mathcal{Z}_{i+1,k-1} \times \mathcal{Z}_{k+1,j} \times E^\omega_{i,k})/\mathcal{Z}_{i,j}$
16 $S_1^\star \leftarrow$ `StructAléatoire`$(i + 1, k - 1, \mathcal{Z}, \omega)$
17 $S_2^\star \leftarrow$ `StructAléatoire`$(k + 1, j, \mathcal{Z}, \omega)$
18 **retourner (** $S_1^\star$ **)** $S_2^\star$

Cas A : Position i sans partenaire dans $S^\star$. On a ici $S^\star = \bullet S'$, où S' est générée sur $[i+1, j]$ avec probabilité de Boltzmann (d'induction), et telle que $E(S^\star) = E(S')$. De plus, la probabilité de choisir ce cas est de $\mathbb{P}(a \leq \mathcal{Z}_{i+1,j}) = \mathcal{Z}_{i+1,j}/\mathcal{Z}_{i,j}$. La probabilité d'émettre $S^\star$ est donc :

$$\mathbb{P}(S^\star) = \frac{\mathcal{Z}_{i+1,j}}{\mathcal{Z}_{i,j}} \cdot \mathbb{P}(S') = \frac{\mathcal{Z}_{i+1,j}}{\mathcal{Z}_{i,j}} \cdot \frac{e^{\frac{-E(S')}{kT}}}{\mathcal{Z}_{i+1,j}} = \frac{e^{\frac{-E(S')}{kT}}}{\mathcal{Z}_{i,j}} = \frac{e^{\frac{-E(S^\star)}{kT}}}{\mathcal{Z}_{i,j}}.$$

Cas B : Positions i et j appariées dans $S^\star$. On a $S^\star = (S')$, où S' est engendrée sur $[i+1, j-1]$ selon Boltzmann (induction), telle que $E(S^\star) = E(S') + E^\omega_{i,k}$. La probabilité de choisir a tel qu'il tombe dans l'intervalle associé à ce cas est de $\mathcal{Z}_{i+1,j-1} \times e^{-E^\omega_{i,j}/kT}/\mathcal{Z}_{i,j}$. La probabilité d'émettre $S^\star$ est donc :

$$\mathbb{P}(S^\star) = \frac{\mathcal{Z}_{i+1,j-1} \times e^{\frac{-E^\omega_{i,j}}{kT}}}{\mathcal{Z}_{i,j}} \times \frac{e^{\frac{-E(S')}{kT}}}{\mathcal{Z}_{i+1,j-1}} = \frac{e^{-\frac{E(S')+E^\omega_{i,j}}{kT}}}{\mathcal{Z}_{i,j}} = \frac{e^{\frac{-E(S^\star)}{kT}}}{\mathcal{Z}_{i,j}}.$$

Cas C : Position i appariée à $k < j$ dans $S^\star$. Dans ce dernier cas, nous avons $S^\star = (S_1)S_2$, où S_1 et S_2 sont engendrés sur les régions $[i+1, k-1]$ et $[k+1, j]$ respectivement, et $E(S^\star) = E^\omega_{i,k} + E(S_1) + E(S_2)$. La probabilité de choisir ce cas est alors

$$\frac{\mathcal{Z}_{i+1,k-1} \times \mathcal{Z}_{k+1,j} \times e^{-E^\omega_{i,k}/kT}}{\mathcal{Z}_{i,j}}$$

et la probabilité d'émission de $S^\star$ est donc de

$$\begin{aligned}\mathbb{P}(S^\star) &= \frac{\mathcal{Z}_{i+1,k-1}\,\mathcal{Z}_{k+1,j}\,e^{\frac{-E^\omega_{i,k}}{kT}}}{\mathcal{Z}_{i,j}} \frac{e^{\frac{-E(S_1)}{kT}}}{\mathcal{Z}_{i+1,k-1}} \frac{e^{\frac{-E(S_2)}{kT}}}{\mathcal{Z}_{k+1,j}} \\ &= \frac{e^{-\frac{E^\omega_{i,k}+E(S_1)+E(S_2)}{kT}}}{\mathcal{Z}_{i,j}} = \frac{e^{-E(S^\star)/kT}}{\mathcal{Z}_{i,j}}.\end{aligned}$$

Comme ces trois cas couvrent exhaustivement et uniquement toutes les structures, la fonction `StructAléatoire`$(i, j, \mathcal{Z}, \omega,)$ retourne donc $S^\star \in \mathcal{S}_{i,j}$ avec la probabilité escomptée. □

1.3.2.2. *Complexité*

En supposant qu'un tirage du nombre aléatoire peut se faire en temps constant, la complexité de l'algorithme `StructAléatoire` est en $\Theta(n^2)$, le cas au pire correspondant à celui analysé à la Section 1.2.1. Il est donc possible d'engendrer un échantillon de M structures en temps $\Theta(n^3 + M.n^2)$.

1.3.2.3. *Pour aller plus loin*

A partir d'un échantillon représentatif de structures, il est possible estimer les propriétés statistiques d'un repliement. On peut ainsi étudier le niveau de structuration d'un ARN à partir de l'espérance du nombre de paires de bases, en partant d'un échantillon de structures aléatoire $S_1, S_2, ..., S_M$ via :

$$\mathbb{E}(\#\text{Paires}(S)) = \frac{\sum_{i=1}^{M} \#\text{Paires}(S_i)}{M}$$

On peut aussi combiner l'échantillonnage avec un algorithme de classification non-supervisée (*clustering*), basée sur une notion de distance des paires de base, pour identifier la ou les conformations dominantes au sein de l'ensemble de Boltzmann (Ding *et al*. 2005).

La complexité moyenne de l'algorithme est en $\Theta(n\sqrt{n})$ (Ponty 2008). Elle peut être améliorée significativement en changeant simplement l'ordre d'examen des valeurs de k dans le cas C. Il suffit pour cela de remplacer l'ordre séquentiel

$$k := i + \theta + 1 \to i + \theta + 2 \rightsquigarrow \ldots \to j - 2 \to j - 1$$

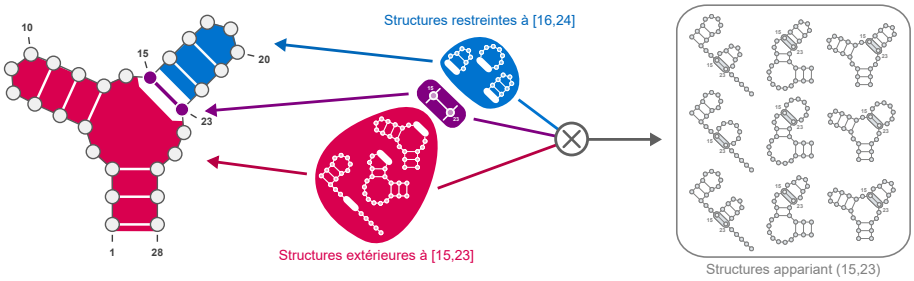


Figure 1.7. *Décomposition des structures faisant apparaître un motif structural, ici la paire de base* $(15, 23)$*. Toute structure contenant* $(15, 23)$ *contient une structure extérieure (rouge), et une partie intérieure (bleu), qui contribuent indépendamment à la stabilité. L'ensemble des structures contenant la paire est obtenu en considérant toutes les combinaisons de structures intérieures et extérieures.*

par un ordre Boustrophedon, allant des extrémités de l'intervalle vers son centre

$$k := i + \theta + 1 \to j - 1 \to i + \theta + 2 \to j - 2 \to \dots$$

Une analyse, hautement technique, de la complexité au pire permet alors de conclure à un temps d'exécution au pire en $\mathcal{O}(n \log n)$ (Ponty 2008).

1.3.3. *Probabilité de Boltzmann d'un motif structural*

Un algorithme d'échantillonnage ouvre à la possibilité d'estimer des propriétés statistiques à l'équilibre thermodynamique, tout en offrant des garanties, sous la forme d'intervalles de confiance, sur la précision des estimations produites. Elle permet par exemple, en engendrant suffisamment de structures, de répondre de façon satisfaisante à des questions telles que : *Quelle est l'énergie moyenne d'un repliement à l'équilibre thermodynamique ?*

Cependant, les intervalles de confiance, basés sur la loi des grands nombres, ne permettent que de contrôler l'erreur absolue. L'échantillonnage rencontre donc des difficultés, ou devient très coûteux, quand on souhaite estimer des quantités de valeur plus faible. En particulier, il n'offre pas de solution satisfaisante à la question : *Quelle sont les probabilités de Boltzmann de toutes les paires de base* (i, j) *?* Plus généralement, on cherche à déterminer la probabilité d'observer des motif structuraux complexes à l'équilibre thermodynamique.

Une contribution majeure de l'article fondateur de McCaskill (1990) permet de calculer de façon efficace la *probabilité de Boltzmann exacte d'un motif structural* m :

$$\mathbb{P}(m \in S) := \sum_{\substack{S \in \mathcal{S} \\ \text{tel que } m \in S}} \mathbb{P}(S \mid w) = \sum_{\substack{S \in \mathcal{S} \\ \text{tel que } m \in S}} \frac{e^{-E(S)/kT}}{\mathcal{Z}} = \frac{\mathcal{Z}_m}{\mathcal{Z}}$$

où $\mathcal{Z}$ est la fonction de partition et $\mathcal{Z}_m := \sum_{S \in \mathcal{S}; m \in S} e^{-E(S)/kT}$ est la fonction de partition restreinte aux structures contenant m. Comme $\mathcal{Z}$ est calculable en temps $\Theta(n^3)$, comme vu en Section 1.3.1, la principale difficulté à trait au calcul de $\mathcal{Z}_m$.

L'algorithme proposé par McCaskill adapte l'approche de l'algorithme *Inside-Outside* (Lari and Young 1990), introduit dans le contexte du traitement automatique des langues. Il repose sur une décomposition non-ambiguë, illustrée à la Figure 1.7, de toutes les structures S contenant $m \in S$ en :

– Une *production* P induite par la décomposition, applicable à une région $[i, j]$, créant une instance du motif m, et suivie d'une ou plusieurs sous-structure(s) sur une/des région(s) $[i_1, j_1], [i_2, j_2] \ldots$;

– Une *structure (intérieure)* S_r sur chaque région $[i_r, j_r]$ issue de P ;

– Une *structure extérieure* S_E à $[i, j]$, c'est à dire une structure laissant un *trou* dans la région $[i, j]$.

Proposition 6. *Soit $E(P)$ la contribution propre d'une production P à l'énergie libre, et $\mathcal{Y}_{i,j}$ la* fonction de partition extérieure *à la région $[i, j]$, telle que*

$$\mathcal{Y}_{i,j} := \sum_{S_E \text{ extérieure à } [i,j]} e^{\frac{-E(S_E)}{kT}} \qquad [1.11]$$

Alors on a :

$$\mathcal{Z}_m = \sum_{\substack{P=([i,j] \to [i_1,j_1],\ldots,[i_r,j_r]) \\ \text{tel que } m \in P}} e^{\frac{-E(P)}{kT}} \mathcal{Y}_{i,j} \prod_r \mathcal{Z}_{i_r,j_r} \qquad [1.12]$$

Démonstration. Remarquons tout d'abord que, pour toute structure S faisant apparaître m, on a $E(S) = E(P) + E(S_E) + \sum_r E(S_r)$. Considérons ensuite la quantité

$$\Phi := \sum_{\substack{P=[i,j] \to [i_1,j_1],[i_2,j_2]\ldots \\ \text{tel que } m \in P}} e^{\frac{-E(P)}{kT}} \mathcal{Y}_{i,j} \prod_r \mathcal{Z}_{i_r,j_r}.$$

En remplaçant les fonctions de partitions par leur définitions respectives, on obtient

$$\Phi = \sum_{P;m\in P} e^{\frac{-E(P)}{kT}} \left(\sum_{\substack{S_E \text{ ext.} \\ \text{à } [i,j]}} e^{\frac{-E(S_E)}{kT}} \right) \prod_r \left(\sum_{\substack{S_r \\ \text{sur } [i_r,j_r]}} e^{\frac{-E(S_r)}{kT}} \right)$$
$$= \sum_{P;m\in P} \sum_{S_E,S_1,S_2,\ldots} e^{-\frac{E(P)+E(S_E)+\sum_r E(S_r)}{kT}} = \sum_{\substack{S\in\mathcal{S} \\ \text{tel que } m\in S}} e^{-E(S)/kT} \equiv \mathcal{Z}_m$$

□

Les fonctions de partition intérieures $\mathcal{Z}_{i,j}$ étant calculables en temps $\Theta(n^3)$, simultanément pour toutes les régions $[i,j]$, l'unique élément manquant pour calculer $\mathcal{Z}_m$, et donc p_m, est la fonction de partition extérieure $\mathcal{Y}$.

Fonction de partition extérieure

Entrée: Séquence $\omega \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{U}\}^+$
Sortie: La *fonction de partition extérieure* $\mathcal{Y}$ associée à chaque région $[i,j]$, définie telle que

$$\mathcal{Y}_{i,j} = \sum_{S_E \text{ extérieure à } [i,j]} e^{-E(S_E)/kT}$$

Heureusement, $\mathcal{Y}$ obéit à une formule relativement simple, basée sur la décomposition décrite dans la Figure 1.8, qu'on établit en *inversant les règles* de production du schéma de programmation dynamique. On obtient alors, pour $i > 1$:

$$\mathcal{Y}_{i,j} = \sum \begin{cases} \mathcal{Y}_{i-1,j} & \blacktriangleright\ \overline{\mathsf{A}} : \text{Pos. } i-1 \text{ est libre} \\ \text{// Si } j < n \text{ et } j-i > \theta : & \blacktriangleright\ \overline{\mathsf{B}} : \text{Paire } (i-1, j+1) \\ \quad e^{\frac{-E^{\omega}_{i-1,j+1}}{kT}}\, \mathcal{Y}_{i-1,j-1} & \\ \text{// Si } j-i > \theta : & \blacktriangleright\ \overline{\mathsf{C}}_{\mathsf{G}} : \text{Paire } (i-1, j+1) \\ \quad \sum_{k=j+2}^{n} e^{\frac{-E^{\omega}_{i-1,j+1}}{kT}}\, \mathcal{Y}_{i-1,k}\, \mathcal{Z}_{k+1,j} & \qquad \text{dans } [i-1, k>j] \\ \sum_{h=1}^{i-\theta-2} e^{\frac{-E^{\omega}_{h,i-1}}{kT}}\, \mathcal{Y}_{h,j}\, \mathcal{Z}_{h+1,i-2} & \blacktriangleright\ \overline{\mathsf{C}}_{\mathsf{D}} : \text{Paire } (h, i-1) \\ & \qquad \text{dans } [h<i, j] \end{cases} \quad [1.13]$$

avec $\mathcal{Y}_{1,n} := 1$ et $\mathcal{Y}_{1,j<n} := 0$. Cette équation peut être calculée pour tout intervalle par programmation dynamique comme décrit dans l'Algorithme 6. Cet algorithme a alors une complexité en temps $\Theta(n^3)$, et en mémoire $\Theta(n^2)$.

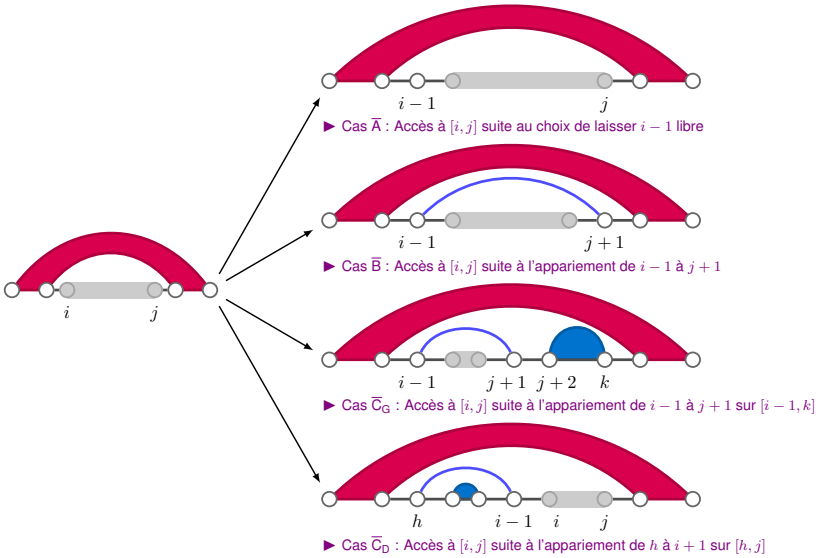


Figure 1.8. *Décomposition des structures extérieures à une région* $[i, j]$.

ALGORITHME 6 : Fonction de partition extérieure

Entrée : ω – ARN de taille n
Sortie : $\mathcal{Z}$ – Matrice $\mathcal{Y}$, remplie selon [1.13]

$\mathcal{Y} \leftarrow$ `MatriceVide`$(n \times n)$ // Initialiser à 0 toutes les valeurs jusqu'à θ de la diagonale.
pour $j \leftarrow 1$ **à** $n-1$ **faire** $\mathcal{Y}_{1,j} \leftarrow 0$
$\mathcal{Y}_{1,n} \leftarrow 1$
pour $i \leftarrow 2$ **à** n **faire**
  pour $j \leftarrow i$ **à** n **faire**
    ► Cas $\overline{\text{A}}$: Position i laissée sans partenaire
    $\mathcal{Y}_{i,j} \leftarrow \mathcal{Y}_{i-1,j}$
    ► Cas $\overline{\text{B}}$: Positions i et j forment une paire de base
    si $j < n$ *et* $j - i > \theta + 2$ **alors**
      $\mathcal{Y}_{i,j} \leftarrow \mathcal{Y}_{i,j} + e^{-E^{\omega}_{i-1,j+1}/kT} \times \mathcal{Y}_{i-1,j-1}$
    ► Cas $\overline{\text{C}_\text{G}}$: Position i appariée à $k < j$
    si $j < n$ *et* $j - i > \theta + 2$ **alors**
      pour $k \leftarrow j+2$ **à** n **faire**
        $\mathcal{Y}_{i,j} \leftarrow \mathcal{Y}_{i,j} + \mathcal{Y}_{i-1,k} \times e^{-E^{\omega}_{i-1,j+1}/kT} \times \mathcal{Z}_{k+1,j}$
    ► Cas $\overline{\text{C}_\text{D}}$: Position i appariée à $k < j$
    pour $h \leftarrow 1$ **à** $i - \theta - 2$ **faire**
      $\mathcal{Y}_{i,j} \leftarrow \mathcal{Y}_{i,j} + \mathcal{Y}_{h,j} \times e^{-E^{\omega}_{h,i-1}/kT} \times \mathcal{Z}_{h+1,i-2}$
retourner $\mathcal{Y}$

Pour calculer la probabilité de Boltzmann d'un motif, il nous reste donc à énumérer les transitions produisant un motif particulier. Concrètement, la probabilité de laisser une position i libre est donnée par

$$\mathbb{P}(i \text{ libre}) = \frac{\mathcal{Y}_{i,i} + \sum_{j=i+1}^{n} \mathcal{Y}_{i,j}\mathcal{Z}_{i+1,j}}{\mathcal{Z}_{1,n}}.$$

De même, la probabilité de former une paire de base (i, j) est obtenue via

$$\mathbb{P}(\text{paire } (i,j)) = \frac{e^{\frac{-E^{\omega}_{i,j}}{kT}}\mathcal{Y}_{i,j}\mathcal{Z}_{i+1,j-1} + \sum_{k=j+1}^{n} e^{\frac{-E^{\omega}_{i,j}}{kT}}\mathcal{Y}_{i,k}\mathcal{Z}_{i+1,j-1}\mathcal{Z}_{j+1,k}}{\mathcal{Z}_{1,n}}.$$

Ces probabilités peuvent typiquement être calculées simultanément pour toutes les positions possibles du motif en temps $\mathcal{O}(n^3)$.

1.3.3.1. *Pour aller plus loin*

Afin de bénéficier d'un algorithme efficace, ici en temps $\mathcal{O}(n^3)$, le motif doit être identifiable dans le schéma de programmation dynamique. Dans la décomposition de Nussinov, cette contrainte limite les motifs éligibles aux paires de bases et positions non-appariées, mais différents types de boucles pourront aussi être considérés au sein de décompositions plus complexes, supportant le modèle d'énergie de Turner comme vu en Section 1.4.1.

Le cadre décrit ci-dessus considère des motifs (paire de base (i, j) donnée, position i non-appariée) qui apparaissent au plus une fois dans chaque structure. On peut cependant utiliser exactement le même calcul pour des motifs apparaissant plusieurs fois (paires de bases, toutes positions confondues), et l'on obtient alors l'*espérance du nombre d'occurrence du motif considéré* dans la distribution de Boltzmann.

1.4. Etudier la structure des ARN en pratique

1.4.1. *Modèle de Turner*

Le modèle d'énergie libre basé sur les paires de bases, utilisé tout au long de ce chapitre, peut sembler un peu simpliste et l'est ! En effet, les paires de base ne représentent pas à les principaux déterminants de l'énergie libre, mais sont dominées par les contributions de "blocs" structuraux (Xia *et al.* 1998) appelés *boucles*, *i.e.* les régions fermées apparaissant dans le dessin de la structure secondaire (voir Figure 1.9). En particulier, les *empilements* de deux paires de bases directement imbriquées $(i, j) \to (i + 1, j - 1)$, sont souvent à l'origine des principales contributions à l'énergie libre. Le énergies associées aux différents types et contenus des boucles, ont été précisément calculées à partir d'expériences de calorimétrie (Turner and Mathews 2010).

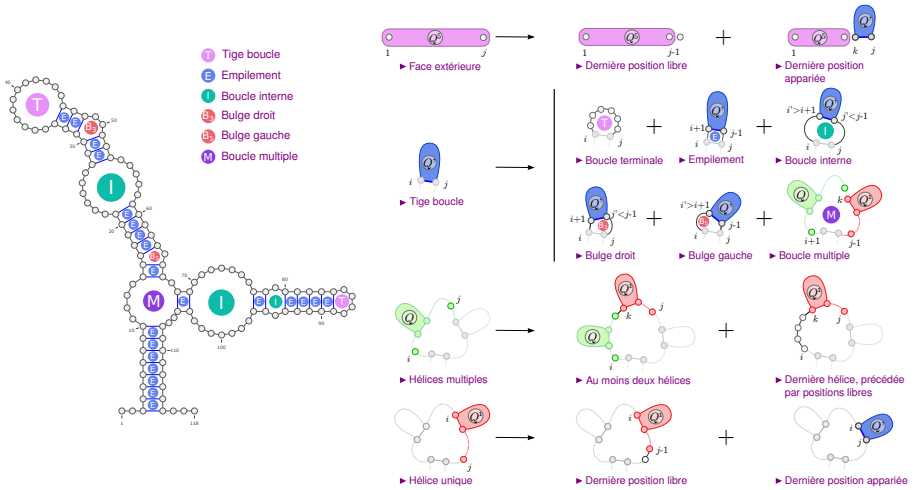
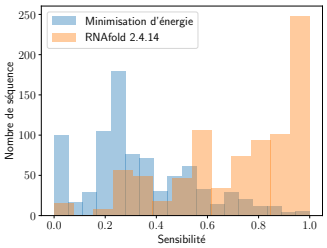


Figure 1.9. *Boucles du modèle de Turner, illustrée sur la structure d'un ARN ribosomal 5s, et décomposition possible des structures secondaires pour ce modèle.*

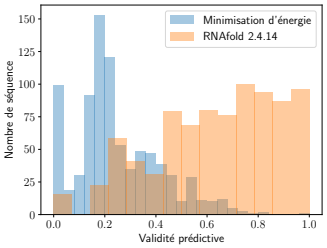
Quoique plus complexe, ce modèle d'énergie préserve une notion d'*indépendance* de motifs locaux dans la structure. Pour cette raison, il peut être pris en compte par une décomposition, illustrée en Figure 1.9, qui préserve les mêmes propriétés (complétude, non-ambiguité, correction) que la version simple présentée dans ce chapitre. Toutes les méthodes et approches algorithmiques illustrées dans ce chapitre peuvent donc être adaptées, essentiellement avec la même complexité, à ce modèle plus réaliste. Le gain de performance qui en résulte est notable, comme l'illustre la Figure 1.10.

1.4.2. *Outils*

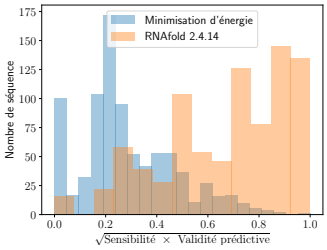
Une multitude d'outils existent afin de replier une séquence d'ARN. Parmi les implémentations les plus populaires, on trouve la suite logicielle `ViennaRNA` (Lorenz *et al*. 2011). Elle contient un vaste ensemble d'options, en plus d'interfaces `Python` et `Perl` permettant une intégration aisée dans un pipeline d'analyse. Ces fonctionnalités en font l'un des outils les plus complets, et fortement populaire en bioinformatique des ARN. Le pacakage `RNAstructure` offre aussi ne nombreuses possibilités, et offre une multitude d'options, ainsi qu'une interface graphique en Java (Reuter and Mathews 2010).



(a) Sensitivité



(b) Validité prédictive



(c) MCC

Figure 1.10. *Sensibilité, validité prédictive, et MCC de la prédiction par minimisation d'énergie, dans notre modèle simplifié (paire de bases) et dans le modèle de Turner.*

1.5. Bibliographie

Ding, Y., Chan, C. Y., Lawrence, C. E. (2005), RNA secondary structure prediction by centroids in a boltzmann weighted ensemble., *RNA (New York, N.Y.)*, 11, 1157–1166.

Ding, Y., Lawrence, C. E. (2003), A statistical sampling algorithm for RNA secondary structure prediction., *Nucleic acids research*, 31, 7280–7301.

Gotoh, O. (1982), An improved algorithm for matching biological sequences, *J Mol Biol*, 162, 705–708.

Huang, L., Chiang, D. (2005), Better k-best parsing, *in* Proceedings of the Ninth International Workshop on Parsing Technology, Association for Computational Linguistics, Vancouver, British Columbia, pp. 53–64.

Kalvari, I., Argasinska, J., Quinones-Olvera, N., Nawrocki, E. P., Rivas, E., Eddy, S. R., Bateman, A., Finn, R. D., Petrov, A. I. (2017), Rfam 13.0 : shifting to a genome-centric resource for non-coding RNA families, *Nucleic Acids Research*, 46(D1), D335–D342.

Tâche	Section	Outil/commande	Package/Ref.
Minimisation d'énergie	1.2.1	`RNAfold`	`ViennaRNA`
		`Fold`	`RNAStructure`
Repliements sous-optimaux	1.2.2	`RNAsubopt`	`ViennaRNA`
		`AllSub`	`RNAStructure`
Repliement comparatif	1.2.3	`LocARNA`	`ViennaRNA`
		`dynalign`	`RNAStructure`
		`FoldAlign`	Sundfeld *et al*. (2015)
Fonction de partition	1.3.1	`RNAfold -p`	`ViennaRNA`
		`partition`	`RNAStructure`
Echantillonnage statistique	1.3.2	`RNAsubopt -p`	`ViennaRNA`
		`stochastic`	`RNAStructure`
Probabilités de Boltzmann	1.3.3	`RNAfold -p`	`ViennaRNA`
		`partition`	`RNAStructure`
Repliement multiple	–	`RNAalifold`	`ViennaRNA`
Annotation structure 3D	–	`DSSR`	Lu *et al*. (2015)

Tableau 1.1. *Implémentations de référence des algorithmes vus dans ce chapitre (modèle d'énergie de Turner).*

Lari, K., Young, S. J. (1990), The estimation of stochastic context-free grammars using the inside-outside algorithm, *Computer Speech and Language*, 4, 35–56.

Lorenz, R., Bernhart, S. H., Höner Zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., Hofacker, I. L. (2011), ViennaRNA package 2.0., *Algorithms for molecular biology : AMB*, 6, 26.

Lu, X.-J., Bussemaker, H. J., Olson, W. K. (2015), DSSR : an integrated software tool for dissecting the spatial structure of RNA, *Nucleic Acids Research*, p. gkv716.

McCaskill, J. S. (1990), The equilibrium partition function and base pair binding probabilities for RNA secondary structure, *Biopolymers : Original Research on Biomolecules*, 29(6-7), 1105–1119.

Michel, F., Westhof, E. (1990), Modelling of the three-dimensional architecture of group i catalytic introns based on comparative sequence analysis, *Journal of Molecular Biology*, 216(3), 585–610.

Mückstein, U., Tafer, H., Hackermüller, J., Bernhart, S. H., Stadler, P. F., Hofacker, I. L. (2006), Thermodynamics of rna-rna binding., *Bioinformatics (Oxford, England)*, 22, 1177–1182.

Nussinov, R., Pieczenik, G., Griggs, J. R., Kleitman, D. J. (1978), Algorithms for loop matchings, *SIAM Journal on Applied mathematics*, 35(1), 68–82.

Ponty, Y. (2008), Efficient sampling of RNA secondary structures from the Boltzmann ensemble of low-energy : The boustrophedon method, *Journal of Mathematical Biology*, 56(1-2), 107–127.

Reuter, J. S., Mathews, D. H. (2010), RNAstructure : software for RNA secondary structure prediction and analysis, *BMC Bioinformatics*, 11(1).

Sankoff, D. (1985), Simultaneous solution of the RNA folding, alignment and protosequence problems, *SIAM Journal on Applied Mathematics*, 45(5), 810–825.

Sundfeld, D., Havgaard, J. H., de Melo, A. C. M. A., Gorodkin, J. (2015), Foldalign 2.5 : multithreaded implementation for pairwise structural RNA alignment, *Bioinformatics*, 32(8), 1238–1240.

Turner, D. H., Mathews, D. H. (2010), NNDB : the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure., *Nucleic acids research*, 38(Database issue), D280–D282.

Wang, L., Jiang, T. (1994), On the complexity of multiple sequence alignment, *Journal of Computational Biology*, 1(4), 337–348.

Waterman, M. (1978), Secondary structure of single-stranded nucleic acids, *Advances in Mathematics : Supplementary Studies*, 1, 167–212.

Waterman, M. S., Byers, T. H. (1985), A dynamic programming algorithm to find all solutions in a neighborhood of the optimum, *Mathematical Biosciences*, 77(1-2), 179–188.

Will, S., Joshi, T., Hofacker, I. L., Stadler, P. F., Backofen, R. (2012), LocaRNA-p : accurate boundary prediction and improved detection of structural RNAs., *RNA (New York, N.Y.)*, 18, 900–914.

Will, S., Otto, C., Miladi, M., Möhl, M., Backofen, R. (2015), Sparse : quadratic time simultaneous alignment and folding of RNAs without sequence-based heuristics., *Bioinformatics (Oxford, England)*, 31, 2489–2496.

Wuchty, S., Fontana, W., Hofacker, I. L., Schuster, P. (1999), Complete suboptimal folding of RNA and the stability of secondary structures, *Biopolymers : Original Research on Biomolecules*, 49(2), 145–165.

Xia, T., SantaLucia Jr, J., Burkard, M. E., Kierzek, R., Schroeder, S. J., Jiao, X., Cox, C., Turner, D. H. (1998), Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson–Crick base pairs, *Biochemistry*, 37(42), 14719–14735.

Zuker, M. (1989), On finding all suboptimal foldings of an rna molecule., *Science (New York, N.Y.)*, 244, 48–52.

Zuker, M., Sankoff, D. (1984), RNA secondary structures and their prediction, *Bulletin of mathematical biology*, 46(4), 591–621.