

Cours M2 BIBS - Séance 1

Repliement *in silico* de l'ARN

Yann Ponty

Bioinformatics Team
École Polytechnique/CNRS/INRIA AMIB – France

<http://www.lix.polytechnique.fr/~ponty/index.php?page=bibscasm2012>

9 Janvier 2012

...ou comment gagner 1 million de dollars en rendant la monnaie!!

Problème : Vous disposez de pièces de **1**, **20** et **50** centimes. Le client souhaite minimiser la monnaie reçue (en nombre de pièces).

Comment rendre **N** en monnaie sans perdre un client ?

Stratégie 1 : Commencer par les *grosses* pièces puis compléter avec les *petites*.

$$21 = ??$$

. . . ou comment gagner 1 million de dollars en rendant la monnaie !!

Problème : Vous disposez de pièces de **1**, **20** et **50** centimes. Le client souhaite minimiser la monnaie reçue (en nombre de pièces).

Comment rendre **N** en monnaie sans perdre un client ?

Stratégie 1 : Commencer par les *grosses* pièces puis compléter avec les *petites*.

$$21 = \text{€}20 + \text{€}1$$

55??

...ou comment gagner 1 million de dollars en rendant la monnaie !!

Problème : Vous disposez de pièces de **1**, **20** et **50** centimes. Le client souhaite minimiser la monnaie reçue (en nombre de pièces).

Comment rendre **N** en monnaie sans perdre un client ?

Stratégie 1 : Commencer par les *grosses* pièces puis compléter avec les *petites*.

$$21 = \text{€20} + \text{€1}$$

$$55 = \text{€50} + \text{€5} + \text{€5} + \text{€5} + \text{€5} + \text{€5}$$

60??

...ou comment gagner 1 million de dollars en rendant la monnaie !!

Problème : Vous disposez de pièces de **1**, **20** et **50** centimes. Le client souhaite minimiser la monnaie reçue (en nombre de pièces).

Comment rendre **N** en monnaie sans perdre un client ?

Stratégie 1 : Commencer par les *grosses* pièces puis compléter avec les *petites*.

$$21 = \text{€}20 + \text{€}1$$

$$55 = \text{€}50 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1$$

$$60 = \text{€}50 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 + \text{€}1 ??$$

...ou comment gagner 1 million de dollars en rendant la monnaie !!

Problème : Vous disposez de pièces de **1**, **20** et **50** centimes. Le client souhaite minimiser la monnaie reçue (en nombre de pièces).

Comment rendre **N** en monnaie sans perdre un client ?

Stratégie 1 : Commencer par les *grosses* pièces puis compléter avec les *petites*.

$$21 = \text{€20} + \text{€1}$$

$$55 = \text{€50} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1}$$

$$60 = \text{€50} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} + \text{€1} ??$$

$$= \text{€20} + \text{€20} + \text{€20} !$$

Problème *a priori* (?!) non-résolvable en général par une approche *gloutonne* car problème plus simple NP-complet (Existe t il même une façon efficace de rendre la monnaie ? \Rightarrow 1M\$).

Stratégie 2 : Il existe une récurrence donnant le nombre minimal de pièce :

$$NbPieces(N) = \text{Min} \begin{cases} \text{€1} & \rightarrow 1 + NbPieces(N - 1) \\ \text{€20} & \rightarrow 1 + NbPieces(N - 20) \\ \text{€50} & \rightarrow 1 + NbPieces(N - 50) \end{cases}$$

Avec un peu de mémoire (N résultats intermédiaires/cas à retenir), on peut alors répondre après $N \times \#Pièces$ calculs.

Remarque : On n'a pas gagné le million, car N a une valeur exponentielle sur son codage. Cet algorithme est donc en temps exponentiel au regard de la théorie de la complexité.

Mais on a optimisé, en évitant un parcours exhaustif de l'arbre des possibles :
 \Rightarrow Programmation dynamique.

1 Introduction

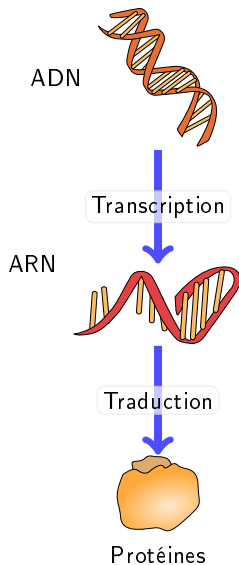
- Fonction(s) de l'ARN
- Repliement et structure
- Représentations de la structure secondaire

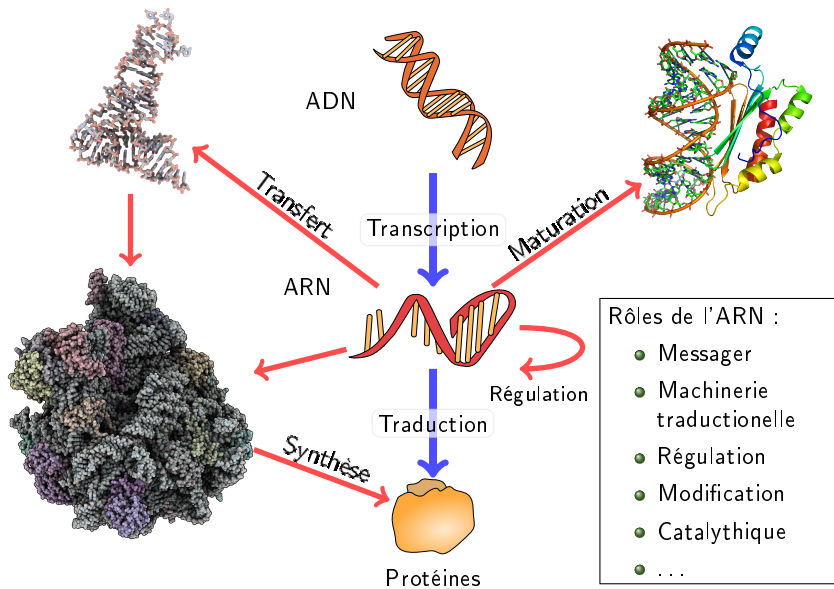
2 Formalisation du repliement et outils disponibles

- Aparté thermodynamique
- Programmation dynamique : Rappels

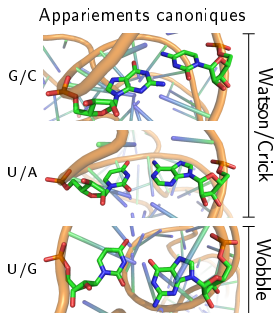
3 Minimisation de l'énergie libre

- Modèle de Nussinov
- Modèle de Turner
- MFold/Unafold
- Performances et approches comparatives
- Vers une prédiction ab-initio 3D





ARN = Biopolymère composé de nucléotides A, C, G et U
A : Adénosine, C : Cytosine, G : Guanine et U : Uracile



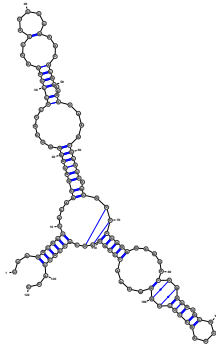
Repliement de l'ARN = Processus stochastique continu dirigé par (résultant en) un appariement des nucléotides.

Comprendre le repliement des ARN aide à comprendre et prédire leur fonction.

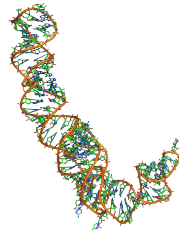
Trois¹ niveaux de représentation :

```
UUAGCGGCCACAGC
GGUGGGGUUGCCUCC
CGUACCCAUCCGAA
CACGGAAGUAAGCC
CACCAGCGUCCGGG
GAGUACUGGAGUGCG
CGAGCCUCUGGGAAA
CCCGGUUCGCCGCCA
CC
```

Structure primaire



Structure secondaire



Structure tertiaire

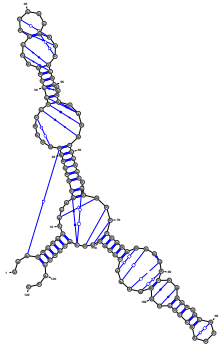
Source : 5s rRNA (PDB 1K73 :B)

1. Enfin, presque ...

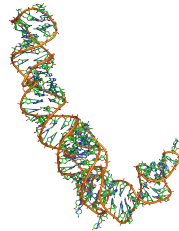
Trois¹ niveaux de représentation :

```
UUAGCGGCCACAGC
GGUGGGGUUGCCUCC
CGUACCCAUCCGAA
CACGGAAGUAAGCC
CACCAGCGUCCGGG
GAGUACUGGAGUGCG
CGAGCCUCUGGGAAA
CCCGGUUCGCCGCCA
CC
```

Structure primaire



Structure secondaire⁺



Structure tertiaire

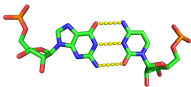
Source : 5s rRNA (PDB 1K73 :B)

1. Enfin, presque ...

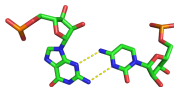
- Appariements non-canoniques

Toute paire de base **autre que** $\{(A-U), (C-G), (G-U)\}$

Ou interagissant sur un bord non-standard (WC/WC-Cis) [LW01].



Paire CG canonique (WC/WC-Cis)



Paire CG non canonique (Sucre/WC-Trans)

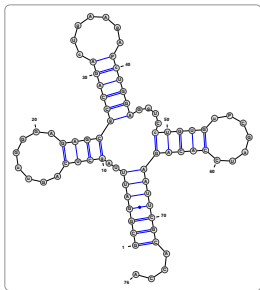
- Pseudonoeuds



Structure pseudonoeud d'un Ribozyme du Groupe I (PDBID : 1Y0Q :A)

Plus expressif, mais repliement général *in silico* avec pseudonoeud :

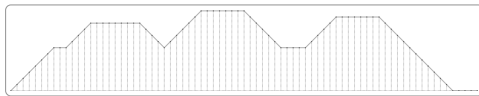
⇒ NP-Complexe [LP00] ... polynomial pour certaines classes [CDR⁺04].



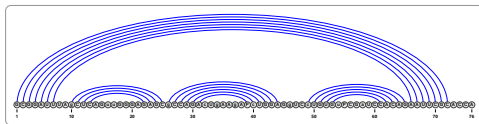
Graphe planaire (outer planar)

((((((.....)))))(((.....))))).....(((.....))))).....

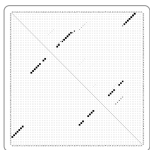
Expression bien parenthésée



Mountain view



Linéaire



Dot plot

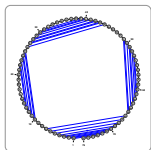


Diagramme de Feynman

Représentation différentes et équivalentes
⇒ Aide l'intuition algorithmique
+ Propriétés algébriques sympathiques
⇒ Algorithmique efficace !

1 Introduction

- Fonction(s) de l'ARN
- Repliement et structure
- Représentations de la structure secondaire

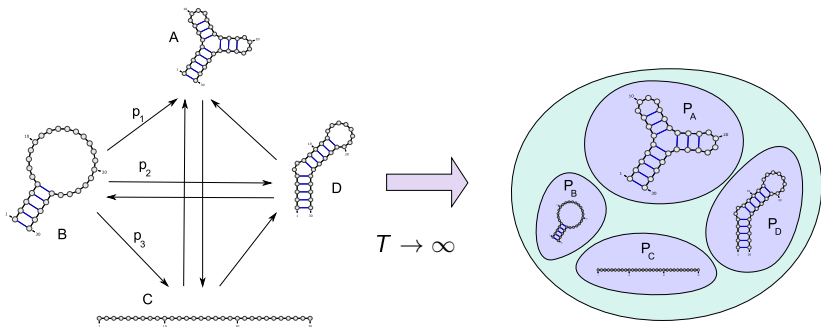
2 Formalisation du repliement et outils disponibles

- Aparté thermodynamique
- Programmation dynamique : Rappels

3 Minimisation de l'énergie libre

- Modèle de Nussinov
- Modèle de Turner
- MFold/Unafold
- Performances et approches comparatives
- Vers une prédiction ab-initio 3D

A l'échelle nanoscopique, la structure de l'ARN *fluctue*.



Convergence vers une **distribution stationnaire** de probabilité, l'équilibre de Boltzmann, où la probabilité est exponentiellement faible sur l'énergie libre.

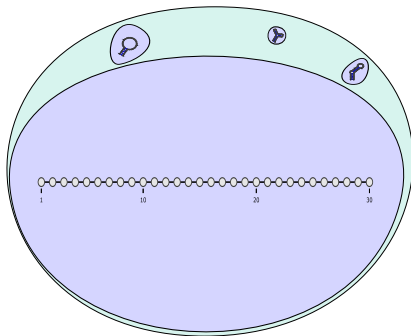
Corollaire : La conformation initiale est sans d'importance.

Problèmes soulevés :

Étant donnés des modèles pour l'ensemble des conformations et l'énergie libre.

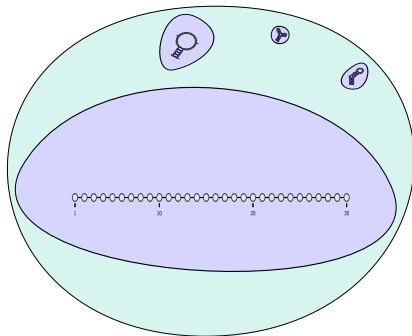
- Déterminer la structure la plus probable (= Energie libre minimale) à l'équilibre
- Déterminer des propriétés moyennes de l'ensemble de Boltzmann

Transcription : ARN synthétisé sans appariement (Sauf exception)



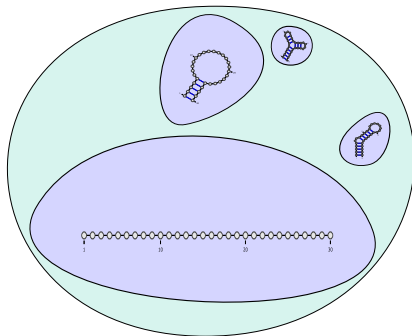
$T = 0$

Transcription : ARN synthétisé sans appariement (Sauf exception)



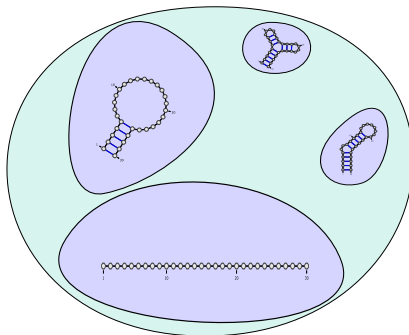
$T = 1h$

Transcription : ARN synthétisé sans appariement (Sauf exception)



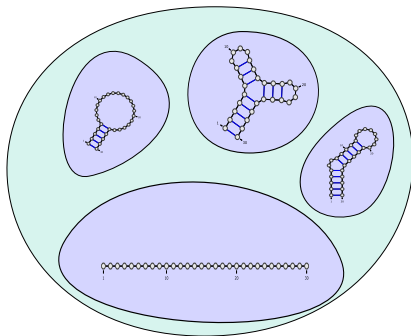
$$T = 2h$$

Transcription : ARN synthétisé sans appariement (Sauf exception)



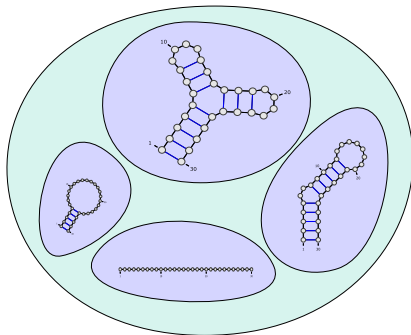
$T = 5h$

Transcription : ARN synthétisé sans appariement (Sauf exception)



$T = 10h$

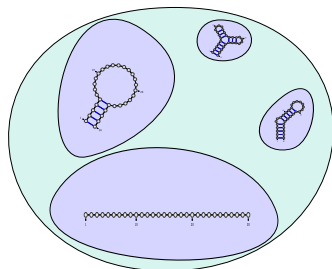
Transcription : ARN synthétisé sans appariement (Sauf exception)



$$T \rightarrow \infty$$

Mais majorité des ARNm dégradés avant 7h (Org. : Souris [SSN⁺09]).

Transcription : ARN synthétisé sans appariement (Sauf exception)



$T = 10h$

Mais majorité des ARNm dégradés avant 7h (Org. : Souris [SSN⁺09]).

- A. Déterminer la structure la plus probable (= Energie libre min.) à l'équilibre
- B. Déterminer des propriétés moyennes de l'ensemble de Boltzmann
- C. Déterminer la structure la plus probable à temps T .
(c.f. H. Isambert par simulation, NP-complet en déterministe [MTSC09])

- 1 Introduction
 - Fonction(s) de l'ARN
 - Repliement et structure
 - Représentations de la structure secondaire
- 2 Formalisation du repliement et outils disponibles
 - Aparté thermodynamique
 - Programmation dynamique : Rappels
- 3 Minimisation de l'énergie libre
 - Modèle de Nussinov
 - Modèle de Turner
 - MFold/Unafold
 - Performances et approches comparatives
 - Vers une prédiction ab-initio 3D

Programmation dynamique = Technique générale pour l'optimisation.

Condition : Solution optimale pour P peut être reconstruite à partir de solutions pour des sous-problèmes strictes de P .

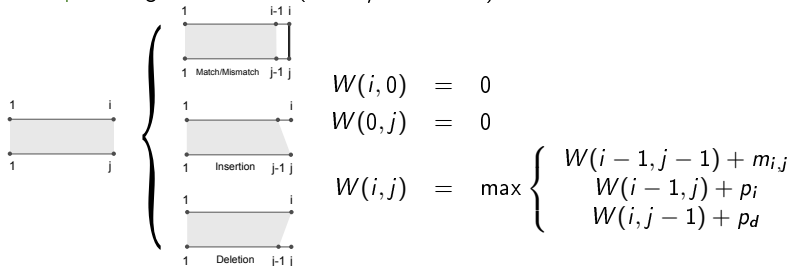
Bioinformatique :

Espace de solutions *discret* (alignements, repliements)

+ Fonction objectif *additive* (score, énergie)

⇒ Schéma de programmation dynamique efficace.

Exemple : Alignement local (Smith/Waterman)



Un schéma fait intervenir des *classes* de sous-problèmes dont on sait calculer le score du *champion*.

Étant donné un schéma, deux étapes :

- **Calcul matrices** : Sauvegarde des meilleurs scores sur classes de sous-problèmes (Ordre inverse de celui induit par les dépendances).
- **Remontée** : Reconstitue le parcours ayant mené au meilleur score. (Parcours = Instance)

Complexité du calcul dépend alors :

- **Taille** de l'espace des sous-problèmes
- **Nombres** de sous-problèmes considérés (#Termes décomposition)

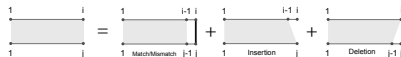
Exemple S/W :

$$i : 1 \rightarrow n + 1 \Rightarrow \Theta(n)$$

$$j : 1 \rightarrow m + 1 \Rightarrow \Theta(m)$$

Trois opération pour chaque sous-calcul

$\Rightarrow \Theta(m.n)$ temps/mémoire



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

	A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0
A	0							
G	0							
C	0							
A	0							
C	0							
A	0							
C	0							
A	0							

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0	0
A	0	→ 2							
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0	0
A	0	2	1						
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

Diagram illustrating the dynamic programming table for local sequence alignment. The table shows the alignment of AGCACACA (row) and ACACACTA (column). The value 2 is highlighted in the cell (A, C), indicating a match. Red arrows point from the 2 to the 1 in the same row and the 0 in the cell (A, A), indicating the path of the alignment.

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_d = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0	0
A	0	2	1	2					
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

Red arrows indicate the path: (0,0) to (1,2) to (1,3) to (1,4). A black arrow points from (1,4) to (0,4).

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1				
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

Diagram illustrating the dynamic programming table for local sequence alignment. The table shows the alignment of AGCACACA (row) and ACACACTA (column). The values in the table represent the maximum score for the alignment up to that point. Red arrows indicate the path of the optimal alignment: (0,0) to (1,1) to (2,2) to (3,3) to (4,4). A grey arrow points from (3,4) to (3,3), and a vertical arrow points from (3,4) to (4,4).

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

Red arrows indicate the path of the local alignment: (0,0) to (1,1) to (1,2) to (2,2) to (2,3) to (3,3) to (3,4) to (4,4) to (4,5) to (4,6) to (4,7) to (4,8) to (4,9).

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0	1	1	1	1	1	1	0	1
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

Diagram illustrating the dynamic programming table for local sequence alignment. The table shows the maximum score $W(i, j)$ for aligning the prefix of sequence 1 (rows) with the prefix of sequence 2 (columns). Red arrows indicate the path of the optimal alignment: A → C → A → C → A → C → T → A.

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0	1	1	1	1	1	1	0	1
C	0	0	3	2	3	2	3	2	1
A	0								
C	0								
A	0								
C	0								
A	0								

Diagram illustrating the dynamic programming table for local sequence alignment. The table shows the maximum score $W(i, j)$ for aligning the prefix of sequence 1 (AGCACACA) up to index i with the prefix of sequence 2 (ACACACTA) up to index j . Red arrows indicate the path of the optimal alignment: (0,0) to (1,1), (1,1) to (2,2), (2,2) to (3,3), (3,3) to (4,4), (4,4) to (5,5), (5,5) to (6,6), (6,6) to (7,7), (7,7) to (8,8), (8,8) to (9,9), (9,9) to (10,10), (10,10) to (11,11), (11,11) to (12,12), (12,12) to (13,13), (13,13) to (14,14), (14,14) to (15,15), (15,15) to (16,16), (16,16) to (17,17), (17,17) to (18,18), (18,18) to (19,19), (19,19) to (20,20).

Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0	1	1	1	1	1	1	0	1
C	0	0	3	2	3	2	3	2	1
A	0	2	2	5	4	5	4	3	4
C	0								
A	0								
C	0								
A	0								

Diagram illustrating the dynamic programming table for local sequence alignment. The table shows the maximum score $W(i, j)$ for aligning the prefix of sequence 1 (rows) with the prefix of sequence 2 (columns). Red arrows indicate the path of the optimal alignment, starting from (0,0) and ending at (5,5), corresponding to the alignment AGCACACA.

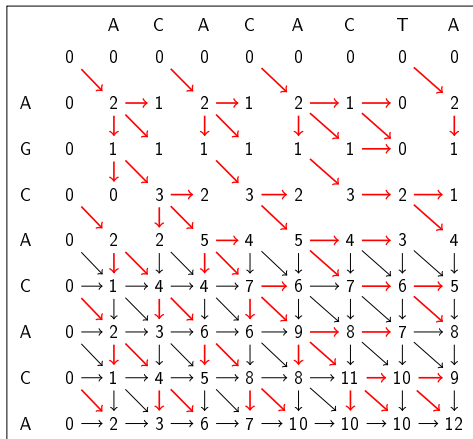
Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

		A	C	A	C	A	C	T	A
0	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0	1	1	1	1	1	1	0	1
C	0	0	3	2	3	2	3	2	1
A	0	2	2	5	4	5	4	3	4
C	0	1	4	4	7	6	7	6	5
A	0	2	3	6	6	9	8	7	8
C	0	1	4	5	8	8	11	10	9
A	0	2	3	6	7	10	10	10	12

Exemple : Alignement local de séquences AGCACACA et ACACACTA

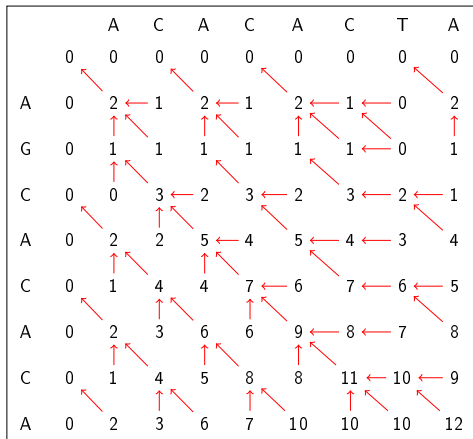
Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

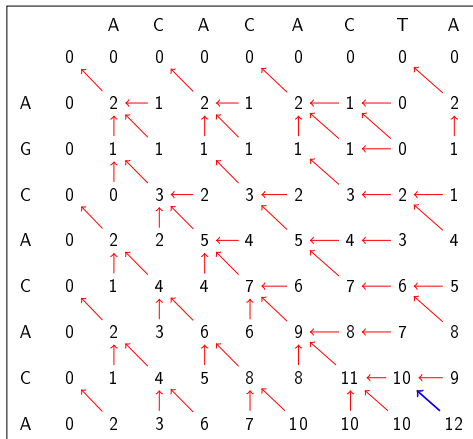
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

A
A



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

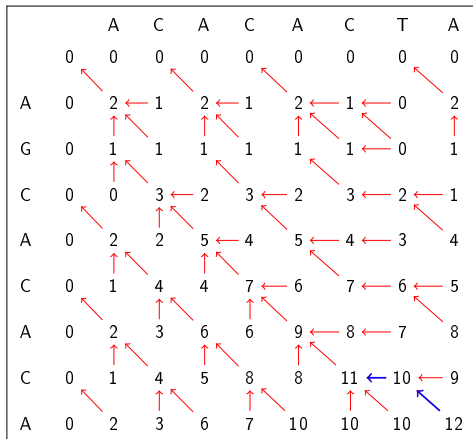
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

- A
T A



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

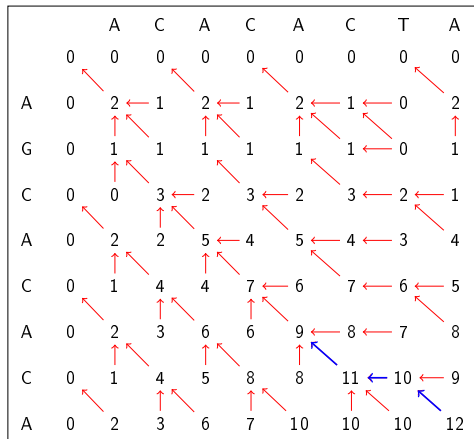
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

C - A
C T A



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

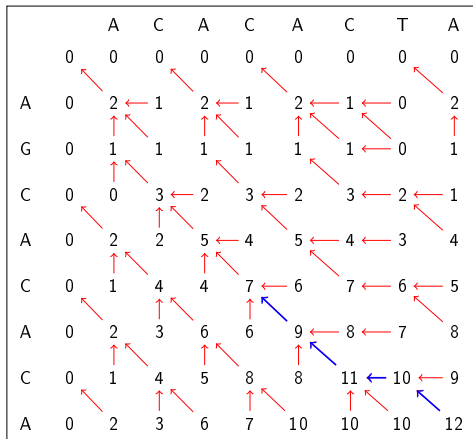
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

A C - A
A C T A



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

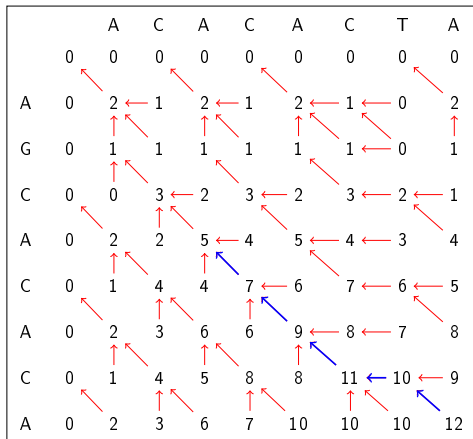
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

C A C - A
C A C T A



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

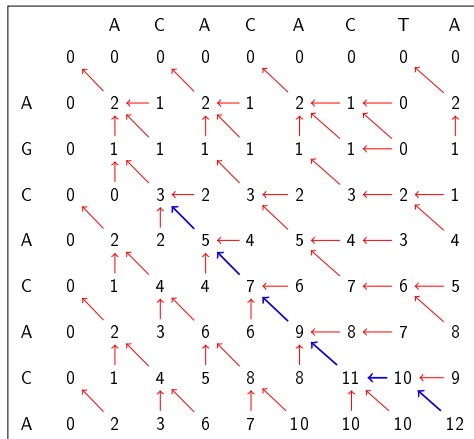
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

A C A C - A
 A C A C T A



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

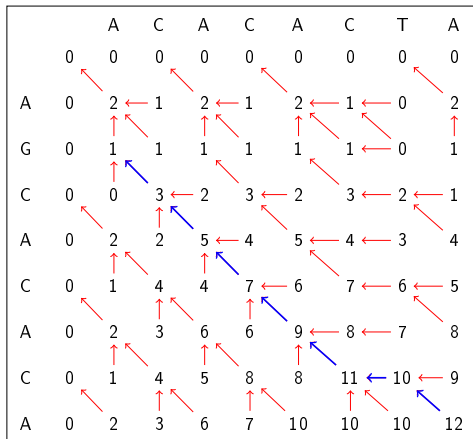
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

C A C A C - A
 C A C A C T A



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

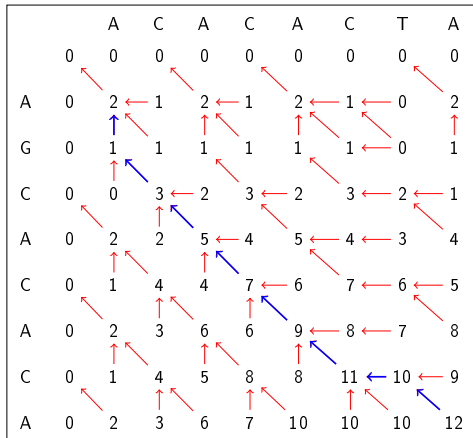
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

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



Exemple : Alignement local de séquences AGCACACA et ACACACTA

Coûts : Match $m_{i,j} = +2$, Insertion/Déletion $p_i = p_j = -1$

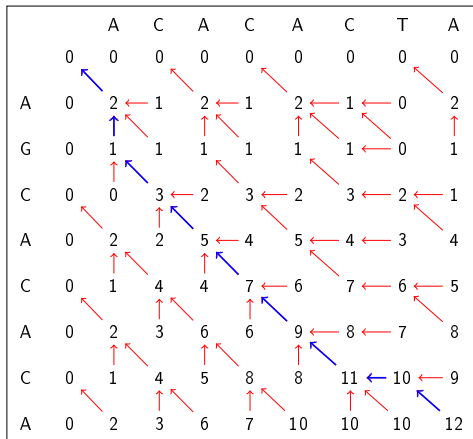
$$W(i, 0) = 0$$

$$W(0, j) = 0$$

$$W(i, j) = \max \begin{cases} W(i-1, j-1) + m_{i,j} \\ W(i-1, j) + p_i \\ W(i, j-1) + p_d \end{cases}$$

Meilleur alignement

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



Propriétés requise d'un schéma :

- **Validité** : \forall sous-problème, la valeur obtenue doit être celle de la fonction objectif.

Preuve souvent assez technique.

Propriétés souhaitables d'un schéma :

- **Complétude** : Espace des solutions engendré par la décomposition.
Des astuces algorithmiques peuvent *couper des branches* . .
- **Non-ambiguïté** : Chaque solution est *engendrée* au plus une fois.

\Rightarrow Possibilité d'**énumérer** l'espace des solutions.

1 Introduction

- Fonction(s) de l'ARN
- Repliement et structure
- Représentations de la structure secondaire

2 Formalisation du repliement et outils disponibles

- Aparté thermodynamique
- Programmation dynamique : Rappels

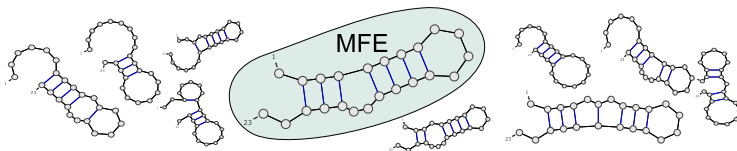
3 Minimisation de l'énergie libre

- Modèle de Nussinov
- Modèle de Turner
- MFold/Unafold
- Performances et approches comparatives
- Vers une prédiction ab-initio 3D

Problème A : Déterminer la structure d'énergie minimale.

Repliement *ab initio* =

Trouver structure d'un ARN ω uniquement à partir de sa séquence.



- **Conformations** : Ensemble S_ω des structures secondaires compatibles avec la structure primaire ω (contrainte d'appariements).
- **Fonction d'énergie** Énergie libre associant une valeur numérique $E_{\omega,S}$ (KCal.mol^{-1}) à tout couple séquence/conformation (ω, S) .
- **Structure native** : Conformation *fonctionnelle* de la molécule.

Remarques :

- Pas nécessairement unique (Cinétique ou structures bi-stables)
- Présence de pseudo-noeuds : Ambiguïté, quelle est la structure native?

Modèle de Nussinov/Jacobson (NJ)

Plus proche voisins simple :

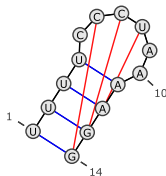
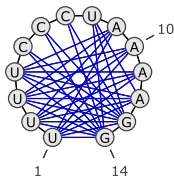
- Seuls les appariements contribuent à l'énergie
- Uniquement liaisons Watson/Crick (A/U,C/G) et Wobble (G/U)

$$\Rightarrow E_{\omega, S} = -\#Paires(S)$$

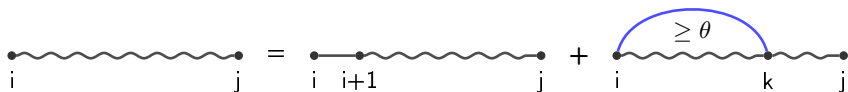
Repliement dans NJ \Leftrightarrow Maximisation du nombre de paires de bases.

Exemple :

UUUUCCCUAAAAGG

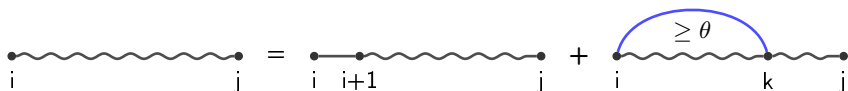


Variante : Pondérer les paires selon leur nombre de liaisons hydrogène
 $\Delta G(G \equiv C) = -3$ $\Delta G(A = U) = -2$ $\Delta G(G - U) = -1$



$$N_{i,t} = 0, \quad \forall t \in [i, i + \theta]$$

$$N_{i,j} = \min \begin{cases} N_{i+1,j} & i \text{ non apparié} \\ \min_{k=i+\theta+1}^j E_{i,k} + N_{i+1,k-1} + N_{k+1,j} & i \text{ apparié à } k \end{cases}$$



$$N_{i,t} = 0, \quad \forall t \in [i, i + \theta]$$

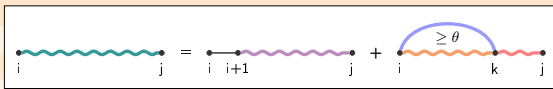
$$N_{i,j} = \min \begin{cases} N_{i+1,j} & i \text{ non apparié} \\ \min_{k=i+\theta+1}^j E_{i,k} + N_{i+1,k-1} + N_{k+1,j} & i \text{ apparié à } k \end{cases}$$

Correction : On cherche à montrer que l'énergie de la structure d'énergie la plus faible ($MFE_{1,n}$) est bien calculée dans $N_{1,n}$. Dans toute structure secondaire restreinte à $[i, j]$ la première position i est :

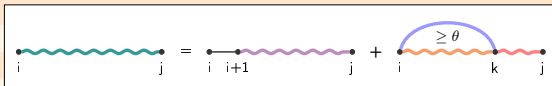
- **Soit non-appariée :** $MFE_{i,j}$ est constituée des appariements de $MFE_{i+1,j}$.
- **Soit appariée à k :** $MFE_{i,j}$ contient l'appariement (i, k) et l'union des appariements de $MFE_{i+1,k-1}$ et de $MFE_{k+1,j}$. En effet, tout appariement entre les régions $[i + 1, k - 1]$ et $[k + 1, j]$ **croiserait** (i, k) (Pseudonoed).

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A

C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10
A						0	0	0	0	0	2	2	2	5	5	5	8	8
C							0	0	0	0	0	0	2	5	5	5	8	8
U								0	0	0	0	0	2	3	5	5	6	7
U									0	0	0	0	2	3	5	5	5	7
C										0	0	0	0	3	3	3	5	5
U											0	0	0	0	2	2	2	3
U												0	0	0	0	0	1	2
A													0	0	0	0	0	0
G														0	0	0	0	0
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0



	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	2	2	4	5	7	7	8	10	10
A						0	0	0	0	2	2	2	5	5	5	8	8	8
C							0	0	0	0	0	2	5	5	5	8	8	8
U								0	0	0	0	2	3	5	5	6	7	7
U									0	0	0	2	3	5	5	5	7	7
C										0	0	0	3	3	3	5	5	5
U											0	0	0	2	2	2	3	3
U												0	0	0	0	1	2	2
A													0	0	0	0	0	0
G														0	0	0	0	0
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0



	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	2	3	5	5	6	7	
U									0	0	0	2	3	5	5	5	7	
C										0	0	0	3	3	3	5	5	
U											0	0	0	2	2	2	3	
U												0	0	0	0	1	2	
A													0	0	0	0	0	
G														0	0	0	0	
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0

$$\text{Sequence}(i, j) = \text{Sequence}(i, i+1) + \text{Sequence}(i+1, j) + \text{Sequence}(i, k) + \text{Sequence}(k, j)$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	2	2	4	5	7	7	8	8	10
A						0	0	0	0	2	2	2	5	5	5	8	8	8
C							0	0	0	0	0	2	5	5	5	8	8	8
U								0	0	0	0	2	3	5	5	6	7	7
U									0	0	0	2	3	5	5	5	7	7
C										0	0	0	3	3	3	5	5	5
U											0	0	0	2	2	2	3	3
U												0	0	0	0	1	2	2
A													0	0	0	0	0	0
G														0	0	0	0	0
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0

$$\text{Sequence } i \dots j = \text{Sequence } i \dots i+1 + \text{Sequence } i+1 \dots j \text{ with pair } (i, k) \geq \theta$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov-Jacobson dynamic programming recurrence. It shows a sequence of nucleotides from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and another from $i+1$ to j . The second part is further partitioned into a subsequence from $i+1$ to k and another from k to j . A blue arc connects nucleotides i and k , labeled with $\geq \theta$, representing a base pair. The diagram is presented as an equation: a sequence from i to j equals a sequence from i to $i+1$ plus a sequence from $i+1$ to j with a base pair between i and k .

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10
A						0	0	0	0	0	2	2	2	5	5	5	8	8
C							0	0	0	0	0	0	2	5	5	5	8	8
U								0	0	0	0	0	2	3	5	5	6	7
U									0	0	0	0	2	3	5	5	5	7
C										0	0	0	0	3	3	3	5	5
U											0	0	0	0	2	2	2	3
U												0	0	0	0	0	1	2
A													0	0	0	0	0	0
G														0	0	0	0	0
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0

Diagram illustrating the Nussinov/Jacobson algorithm for RNA secondary structure prediction. It shows a sequence from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects index i to index k , where $k \geq i+1$. The diagram is represented as: a wavy line from i to j = a wavy line from i to $i+1$ + a wavy line from $i+1$ to j with a blue arc from i to k labeled $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm for RNA secondary structure prediction. It shows a sequence from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects index i to index k , where $k \geq i+1$. The diagram is represented as: a wavy line from i to j = a wavy line from i to $i+1$ + a wavy line from $i+1$ to j with a blue arc from i to k labeled $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov-Jacobson dynamic programming recurrence. It shows a sequence of nucleotides from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . The second part is further partitioned into a subsequence from $i+1$ to k and a subsequence from k to j . A blue arc connects nucleotide i to nucleotide k , representing a base pair. The angle between the arc and the horizontal line is labeled as $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

$$\text{Sequence}(i, j) = \text{Sequence}(i, i+1) + \text{Sequence}(i+1, j) + \sum_{i+1 \leq k < j} \text{Sequence}(i, k) + \text{Sequence}(k, j)$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson dynamic programming recurrence. It shows a sequence of nucleotides from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects nucleotide i to nucleotide k , representing a base pair with energy $\geq \theta$. The equation is: sequence($i..j$) = sequence($i..i+1$) + sequence($i+1..j$) + energy($i..k$).

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

$$\text{Sequence } (i, \dots, j) = \text{Sequence } (i, \dots, i+1) + \text{Sequence } (i+1, \dots, j) \text{ with base pair } (i, k) \geq \theta$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm for RNA folding. It shows a sequence from index i to j . The first part is a wavy line representing a substructure. This is equal to the sum of two terms: 1) a straight line from i to $i+1$ followed by a wavy line from $i+1$ to j , and 2) a wavy line from i to k followed by a curved line from k to j labeled with $\geq \theta$, representing a loop.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
	(.)	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10
A						0	0	0	0	0	2	2	2	5	5	5	8	8
C							0	0	0	0	0	0	2	5	5	5	8	8
U								0	0	0	0	0	2	3	5	5	6	7
U									0	0	0	0	2	3	5	5	5	7
C										0	0	0	0	3	3	3	5	5
U											0	0	0	0	2	2	2	3
U												0	0	0	0	0	1	2
A													0	0	0	0	0	0
G														0	0	0	0	0
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0

$$\text{Sequence } i \dots j = \text{Sequence } i \dots i+1 + \text{Sequence } i \dots k + \text{Loop } (k, j) + \text{Sequence } k \dots j$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	((.))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

$$\text{wavy}(i, j) = \text{wavy}(i, i+1) + \text{wavy}(i+1, j) + \sum_{i < k < j} \text{wavy}(i, k) + \text{wavy}(k, j) \text{ if } \theta \leq \text{score}(i, k, j)$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
	((.))	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10
A						0	0	0	0	0	2	2	2	5	5	5	8	8
C							0	0	0	0	0	0	2	5	5	5	8	8
U								0	0	0	0	0	2	3	5	5	6	7
U									0	0	0	0	2	3	5	5	5	7
C										0	0	0	0	3	3	3	5	5
U											0	0	0	0	2	2	2	3
U												0	0	0	0	0	1	2
A													0	0	0	0	0	0
G														0	0	0	0	0
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0

Diagram illustrating the Nussinov/Jacobson algorithm for RNA secondary structure prediction. It shows a sequence from index i to j . The first part is a wavy line representing a substructure. This is equal to the sum of two terms: 1) a wavy line from i to $i+1$ followed by a wavy line from $i+1$ to j , and 2) a wavy line from i to k followed by a loop (arc) from k to i with a label $\geq \theta$, and then a wavy line from k to j .

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
	((.))	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10
A						0	0	0	0	0	2	2	2	5	5	5	8	8
C							0	0	0	0	0	0	2	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7
U									0	0	0	0	2	3	5	5	5	7
C										0	0	0	0	3	3	3	5	5
U											0	0	0	0	2	2	2	3
U												0	0	0	0	0	1	2
A													0	0	0	0	0	0
G														0	0	0	0	0
A															0	0	0	0
C																0	0	0
G																	0	0
A																		0

$i \text{---} j = i \text{---} i+1 + i+1 \text{---} j$

$i \text{---} j \text{ (with arc } i \rightarrow k \text{ where } k \geq i+1 \text{)}$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

$$\text{Sequence } i \dots j = \text{Sequence } i \dots i+1 + \text{Sequence } i \dots k \text{ (with arc } k \dots j \text{)} + \text{Sequence } k \dots j$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A
	(((.	.	.)))	.
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10
U					0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	2	3	5	5	6	7	
U									0	0	0	2	3	5	5	5	7	
C										0	0	0	3	3	3	5	5	
U											0	0	0	2	2	2	3	
U												0	0	0	0	1	2	
A													0	0	0	0	0	
G														0	0	0	0	
A															0	0	0	
C																0	0	
G																	0	
A																		0

$$\text{Sequence } [i, j] = \text{Sequence } [i, i+1] + \text{Sequence } [i+1, j] + \text{Sequence } [i, k] + \text{Sequence } [k, j]$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm for RNA secondary structure prediction. It shows a sequence from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects index i to index k , where $k \geq i+1$. The diagram is represented as: a wavy line from i to j = a wavy line from i to $i+1$ + a wavy line from $i+1$ to j with a blue arc from i to k labeled $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	8	8		
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

$$\text{Sequence } i \dots j = \text{Sequence } i \dots i+1 + \text{Sequence } i \dots k \text{ (with loop } \geq \theta \text{)} + \text{Sequence } k \dots j$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	2	2	4	5	7	7	8	10		
A						0	0	0	0	2	2	2	5	5	5	8	8		
C							0	0	0	0	0	2	5	5	5	8	8		
U								0	0	0	0	2	3	5	5	6	7		
U									0	0	0	2	3	5	5	7			
C										0	0	0	3	3	3	5	5		
U											0	0	2	2	2	3			
U												0	0	0	0	1	2		
A													0	0	0	0	0		
G														0	0	0	0		
A															0	0	0		
C																0	0		
G																	0		
A																		0	

$$\text{Sequence } i \dots j = \text{Sequence } i \dots i+1 + \text{Sequence } i \dots k \text{ (with arc } k \dots j \text{)} + \text{Sequence } k \dots j$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	2	2	4	5	7	7	8	10		
A						0	0	0	0	2	2	2	5	5	5	8	8		
C							0	0	0	0	0	2	5	5	5	8	8		
U								0	0	0	0	2	3	5	5	6	7		
U									0	0	0	2	3	5	5	7			
C										0	0	0	3	3	3	5	5		
U											0	0	0	2	2	2	3		
U												0	0	0	0	1	2		
A													0	0	0	0	0		
G														0	0	0	0		
A															0	0	0		
C																0	0		
G																	0		
A																		0	

$$\text{Path}(i, j) = \text{Path}(i, i+1, j) + \text{Path}(i, k, j) \text{ with } \text{loop} \geq \theta$$

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm for RNA secondary structure prediction. It shows a sequence from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects index i to index k , where k is between $i+1$ and j . The energy contribution of this arc is labeled as $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)	.	(.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm for RNA secondary structure prediction. It shows a sequence from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects index i to index k , where $k \geq i+1$. The diagram is represented as: a wavy line from i to j = a wavy line from i to $i+1$ + a wavy line from $i+1$ to j with a blue arc from i to k labeled $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)	.	(.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm for RNA secondary structure prediction. It shows a sequence from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects index i to index k , where $k \geq i+1$. The diagram is represented as: a wavy line from i to j = a wavy line from i to $i+1$ + a wavy line from $i+1$ to j with a blue arc from i to k labeled $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)	.	(.)))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm. It shows a sequence of nucleotides from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects nucleotide i to nucleotide k , with the label $\geq \theta$, indicating a base pair with energy at least θ .

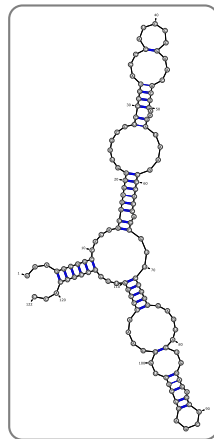
	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)	.	((.	.	.))))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

Diagram illustrating the Nussinov/Jacobson algorithm. It shows a sequence of nucleotides from index i to j . The sequence is partitioned into two parts: a subsequence from i to $i+1$ and a subsequence from $i+1$ to j . A blue arc connects nucleotide i to nucleotide k , where $k \geq i+1$. The diagram is represented as: a wavy line from i to j = a wavy line from i to $i+1$ + a wavy line from $i+1$ to j with a blue arc from i to k labeled $\geq \theta$.

	C	G	G	A	U	A	C	U	U	C	U	U	A	G	A	C	G	A	
	(((.	.	.)	.	((.	.	.))))	.	
C	0	0	0	0	0	0	3	4	4	6	6	6	6	9	9	11	14	14	
G		0	0	0	0	0	3	4	4	6	6	6	6	7	9	11	11	11	
G			0	0	0	0	3	3	3	5	5	5	5	6	8	10	10	10	
A				0	0	0	0	2	2	2	2	4	4	5	7	7	8	10	
U					0	0	0	0	0	0	2	2	4	5	7	7	8	10	
A						0	0	0	0	0	2	2	2	5	5	5	8	8	
C							0	0	0	0	0	0	2	5	5	5	8	8	
U								0	0	0	0	0	2	3	5	5	6	7	
U									0	0	0	0	2	3	5	5	5	7	
C										0	0	0	0	3	3	3	5	5	
U											0	0	0	0	2	2	2	3	
U												0	0	0	0	0	1	2	
A													0	0	0	0	0	0	
G														0	0	0	0	0	
A															0	0	0	0	
C																0	0	0	
G																	0	0	
A																		0	

$$\text{wavy}(i, j) = \text{wavy}(i, i+1) + \text{wavy}(i+1, k) + \text{loop}(i, k, j)$$

Basée sur décomposition **non-ambiguë** en **boucles** de la structure 2^{aire} :



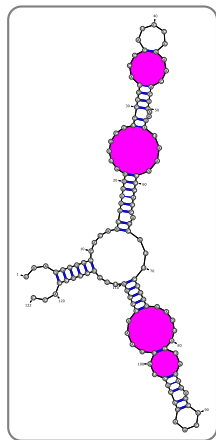
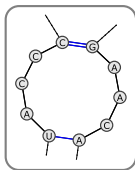
Énergies libres ΔG des boucles dépendent des bases, asymétrie, bases *libres* (dangle) ...

Déterminées expérimentalement
+ Interpolation pour grandes boucles.

Meilleure résultats grâce à la prise en compte de l'empilement.

Basée sur décomposition **non-ambiguë** en **boucles** de la structure 2^{aire} :

- Boucles internes



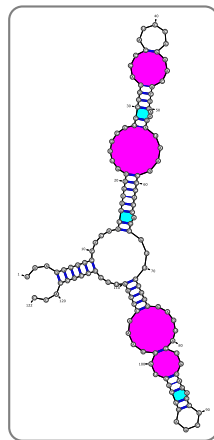
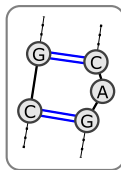
Énergies libres ΔG des boucles dépendent des bases, asymétrie, bases *libres* (dangle) ...

Déterminées expérimentalement
+ Interpolation pour grandes boucles.

Meilleure résultats grâce à la prise en compte de l'empilement.

Basée sur décomposition **non-ambiguë** en **boucles** de la structure 2^{aire} :

- Boucles internes
- Renflements



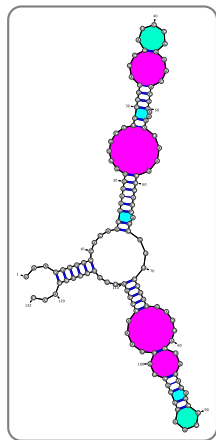
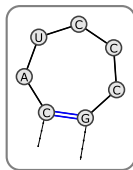
Énergies libres ΔG des boucles dépendent des bases, asymétrie, bases *libres* (dangle) ...

Déterminées expérimentalement
+ Interpolation pour grandes boucles.

Meilleure résultats grâce à la prise en compte de l'empilement.

Basée sur décomposition **non-ambiguë** en **boucles** de la structure 2^{aire} :

- Boucles internes
- Renflements
- Boucles terminales



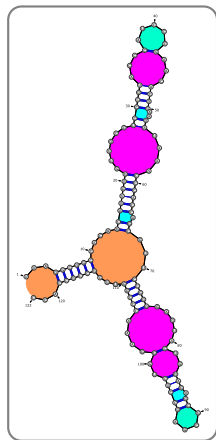
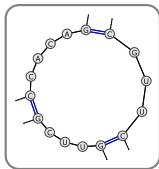
Énergies libres ΔG des boucles dépendent des bases, asymétrie, bases *libres* (dangle) ...

Déterminées expérimentalement
+ Interpolation pour grandes boucles.

Meilleure résultats grâce à la prise en compte de l'empilement.

Basée sur décomposition **non-ambiguë** en **boucles** de la structure 2^{aire} :

- Boucles internes
- Renflements
- Boucles terminales
- Boucles multiples



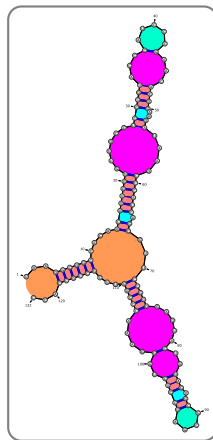
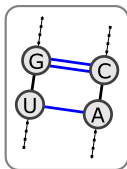
Énergies libres ΔG des boucles dépendent des bases, asymétrie, bases *libres* (dangle) ...

Déterminées expérimentalement
+ Interpolation pour grandes boucles.

Meilleure résultats grâce à la prise en compte de l'empilement.

Basée sur décomposition **non-ambiguë** en **boucles** de la structure 2^{aire} :

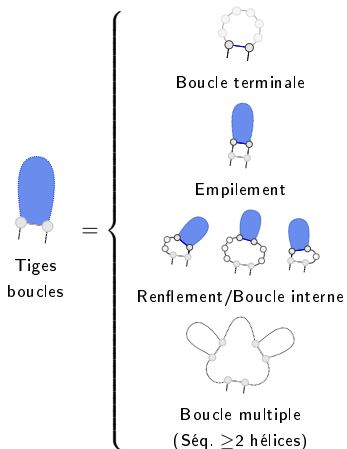
- Boucles internes
- Renflements
- Boucles terminales
- Boucles multiples
- Empilements

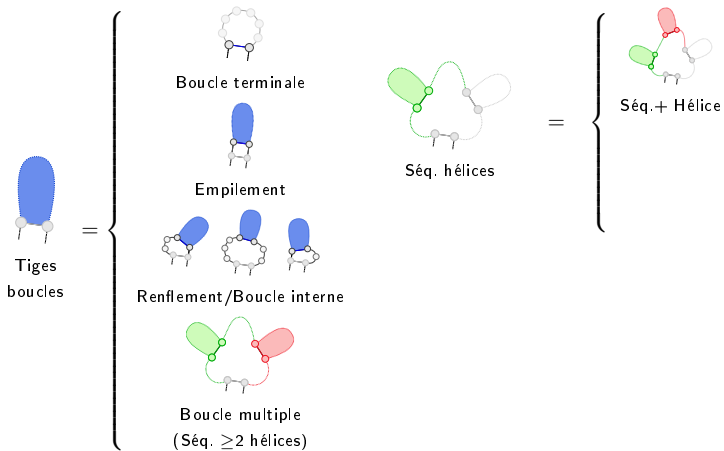


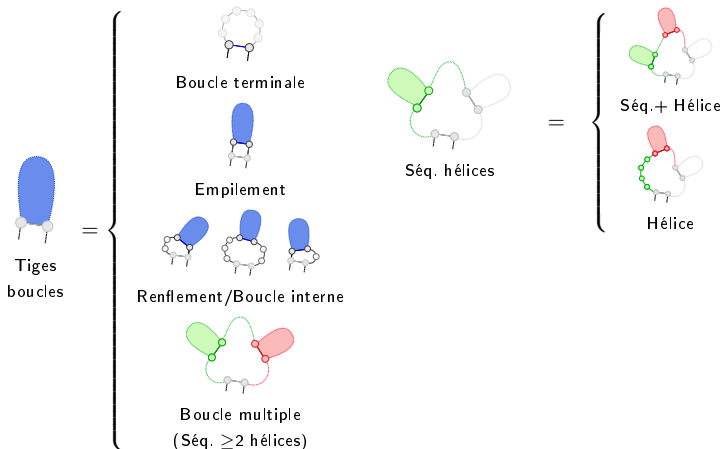
Énergies libres ΔG des boucles dépendent des bases, asymétrie, bases *libres* (dangle) ...

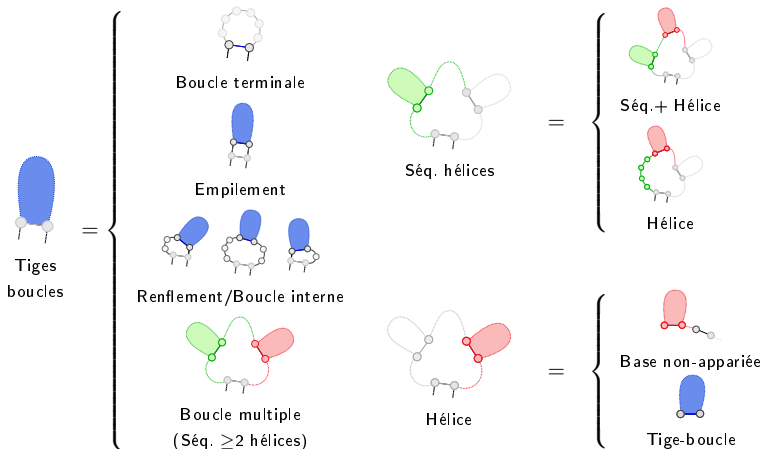
Déterminées expérimentalement
+ Interpolation pour grandes boucles.

Meilleure résultats grâce à la prise en compte de l'empilement.

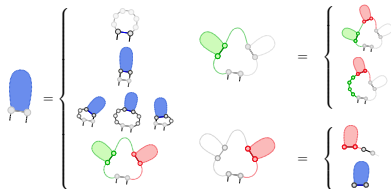








- $E_H(i, j)$: Energie de boucle terminale *fermée* par une paire (i, j)
- $E_{BI}(i, j)$: Energie de renflement ou boucle interne *fermée* par une paire (i, j)
- $E_S(i, j)$: Energie d'empilement $(i, j)/(i + 1, j - 1)$
- a, c, b : Pénalité de boucle multiple, hélice et non-appariées dans multiboucle.



Calcul des matrices

$$\begin{aligned}
 \mathcal{M}'_{i,j} &= \min \begin{cases} E_H(i, j) \\ E_S(i, j) + \mathcal{M}'_{i+1, j-1} \\ \text{Min}_{i', j'} (E_{BI}(i, i', j', j) + \mathcal{M}'_{i', j'}) \\ a + c + \text{Min}_k (\mathcal{M}_{i+1, k-1} + \mathcal{M}^1_{k, j-1}) \end{cases} \\
 \mathcal{M}_{i,j} &= \text{Min}_k \{ \min (\mathcal{M}_{i, k-1}, b(k-1)) + \mathcal{M}^1_{k, j} \} \\
 \mathcal{M}^1_{i,j} &= \text{Min}_k \{ b + \mathcal{M}^1_{i, j-1}, c + \mathcal{M}'_{i, j} \}
 \end{aligned}$$

Reconstruction de la structure d'énergie minimale :

$$\mathcal{M}'_{i,j} = \text{Min} \left\{ \begin{array}{l} E_H(i,j) \\ E_S(i,j) + \mathcal{M}'_{i+1,j-1} \\ \text{Min}_{i',j'} (E_{BI}(i,i',j',j) + \mathcal{M}'_{i',j'}) \\ a + c + \text{Min}_k (\mathcal{M}_{i+1,k-1} + \mathcal{M}^1_{k,j-1}) \end{array} \right\}$$

$$\mathcal{M}_{i,j} = \text{Min}_k \{ \min (\mathcal{M}_{i,k-1}, b(k-1)) + \mathcal{M}^1_{k,j} \}$$

$$\mathcal{M}^1_{i,j} = \text{Min}_k \{ b + \mathcal{M}^1_{i,j-1}, c + \mathcal{M}'_{i,j} \}$$

2. Avec une astuce pour les bulges/boucles internes ...

Reconstruction de la structure d'énergie minimale :

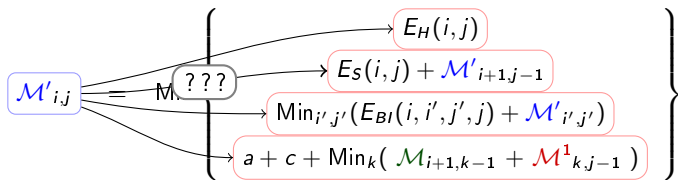
$$\mathcal{M}'_{i,j} \leftarrow \text{Min} \left\{ \begin{array}{l} E_H(i,j) \\ E_S(i,j) + \mathcal{M}'_{i+1,j-1} \\ \text{Min}_{i',j'} (E_{BI}(i,i',j',j) + \mathcal{M}'_{i',j'}) \\ a + c + \text{Min}_k (\mathcal{M}_{i+1,k-1} + \mathcal{M}^1_{k,j-1}) \end{array} \right\}$$

$$\mathcal{M}_{i,j} = \text{Min}_k \{ \min(\mathcal{M}_{i,k-1}, b(k-1)) + \mathcal{M}^1_{k,j} \}$$

$$\mathcal{M}^1_{i,j} = \text{Min}_k \{ b + \mathcal{M}^1_{i,j-1}, c + \mathcal{M}'_{i,j} \}$$

2. Avec une astuce pour les bulges/boucles internes ...

Reconstruction de la structure d'énergie minimale :



$$\mathcal{M}_{i,j} = \text{Min}_k \{ \min(\mathcal{M}_{i,k-1}, b(k-1)) + \mathcal{M}^1_{k,j} \}$$

$$\mathcal{M}^1_{i,j} = \text{Min}_k \{ b + \mathcal{M}^1_{i,j-1}, c + \mathcal{M}'_{i,j} \}$$

$\mathcal{O}(n)$ contributeurs potentiels au Min :

⇒ Complexité au pire en $\mathcal{O}(n^2)$ pour un backtrack naïf.

Reconstruction de la structure d'énergie minimale :

$$\mathcal{M}'_{i,j} = \text{Min} \left\{ \begin{array}{l} E_H(i,j) \\ E_S(i,j) + \mathcal{M}'_{i+1,j-1} \\ \text{Min}_{i',j'} (E_{BI}(i,i',j',j) + \mathcal{M}'_{i',j'}) \\ a + c + \text{Min}_k (\mathcal{M}_{i+1,k-1} + \mathcal{M}^1_{k,j-1}) \end{array} \right\}$$

$$\mathcal{M}_{i,j} = \text{Min}_k \{ \min (\mathcal{M}_{i,k-1}, b(k-1)) + \mathcal{M}^1_{k,j} \}$$

$$\mathcal{M}^1_{i,j} = \text{Min}_k \{ b + \mathcal{M}^1_{i,j-1}, c + \mathcal{M}'_{i,j} \}$$

$\mathcal{O}(n)$ contributeurs potentiels au Min :

⇒ Complexité au pire en $\mathcal{O}(n^2)$ pour un backtrack naïf.

Garder les meilleures contributions aux Min ⇒ Backtrack en $\mathcal{O}(n)$

Complexités temps/mémoire en $\mathcal{O}(n^3)/\mathcal{O}(n^2)$ pour le précalcul²

2. Avec une astuce pour les bulges/boucles internes ...

Reconstruction de la structure d'énergie minimale :

$$\mathcal{M}'_{i,j} = \text{Min} \left\{ \begin{array}{l} E_H(i,j) \\ E_S(i,j) + \mathcal{M}'_{i+1,j-1} \\ \text{Min}_{i',j'} (E_{BI}(i,i',j',j) + \mathcal{M}'_{i',j'}) \\ a + c + \text{Min}_k (\mathcal{M}_{i+1,k-1} + \mathcal{M}^1_{k,j-1}) \end{array} \right\}$$

$$\mathcal{M}_{i,j} \leftarrow - \text{Min}_k \{ \min(\mathcal{M}_{i,k-1}, b(k-1)) + \mathcal{M}^1_{k,j} \}$$

$$\mathcal{M}^1_{i,j} \leftarrow - \text{Min}_k \{ b + \mathcal{M}^1_{i,j-1}, c + \mathcal{M}'_{i,j} \}$$

$\mathcal{O}(n)$ contributeurs potentiels au Min :

⇒ Complexité au pire en $\mathcal{O}(n^2)$ pour un backtrack naïf.

Garder les meilleures contributions aux Min ⇒ Backtrack en $\mathcal{O}(n)$

Complexités temps/mémoire en $\mathcal{O}(n^3)/\mathcal{O}(n^2)$ pour le précalcul²

⇒ **UnaFold** [MZ08] calcule la structure secondaire d'énergie minimale.

2. Avec une astuce pour les bulges/boucles internes ...

Definition (Repliement ab initio)

Partant de la séquence, trouver la conformation minimisant une fonction d'énergie.

Avantages :

- Explication mécanique
- Complexité raisonnable
 $\mathcal{O}(n^3)/\mathcal{O}(n^2)$ temps/mémoire
- Exploration exhaustive

Limites :

- Pas de cinétique
- Pas d'info évolutive
- Performances limitées

Definition (Approche comparative)

Partant de plusieurs séquences homologues ou d'un alignement, trouver une conformation de score (énergie+alignement) élevé.

Avantages :

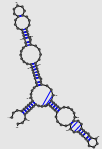
- Meilleures performances
- Affinement permanent

Limites :

- Complexité élevée
- Exploration non-exhaustive

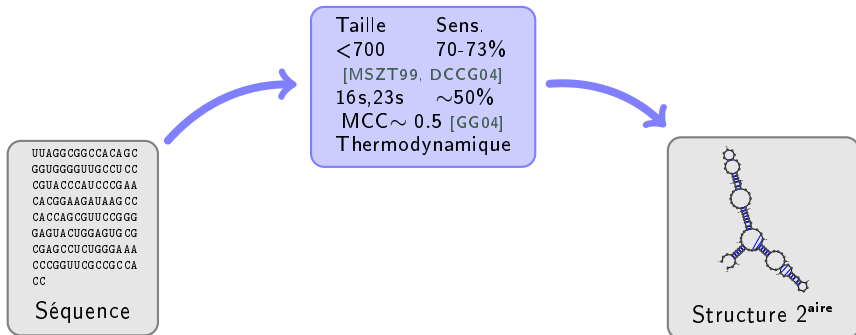
```
UUAGGCGGCCACGC  
GGUGGGUUGCCUCC  
CGUACCCAUCCGAA  
CACGGAAGAUAGCC  
CACCCAGCUUCGGG  
GAGUACUGGAGUCG  
CGAGCCUCUGGGAAA  
CCCGGUUCGCCCA  
CC
```

Séquence

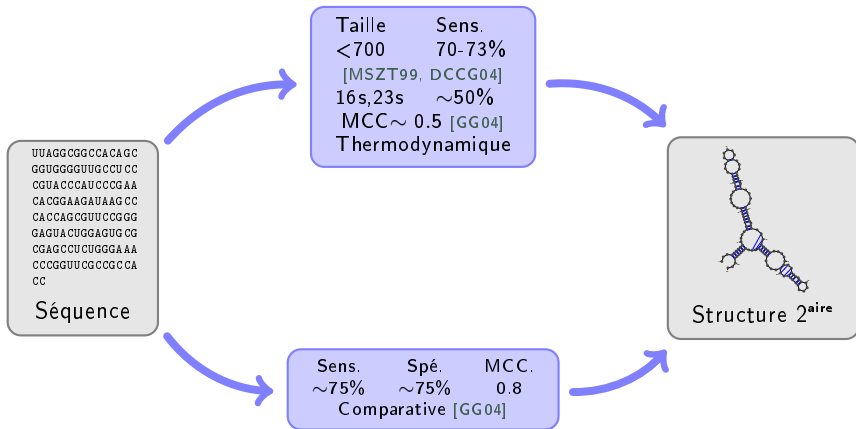


Structure 2^{aire}

$$\text{Rappel : } MCC = \frac{e^+ e^- - f^+ f^-}{\sqrt{(e^+ + f^+)(e^+ + f^-)(e^- + f^+)(e^- + f^-)}}$$



$$\text{Rappel : } MCC = \frac{t^+t^- - f^+f^-}{\sqrt{(t^++f^+)(t^++f^-)(t^-+f^+)(t^-+f^-)}}$$



Rappel :
$$MCC = \frac{e^+e^- - f^+f^-}{\sqrt{(e^++f^+)(e^++f^-)(e^-+f^+)(e^-+f^-)}}$$

But : De la séquence à des modèles tri-dimensionnels!!!

```
UUAGGCGGCCACGC  
GGUGGGGUUGCCUCC  
CGUACCCAUCCGAA  
CACGGAAGAUAGCC  
CACCAAGCUUCCGGG  
GAGUACUGGAGUGCG  
CGAGCCUCUGGGAAA  
CCCGGUUCGCCCA  
CC
```

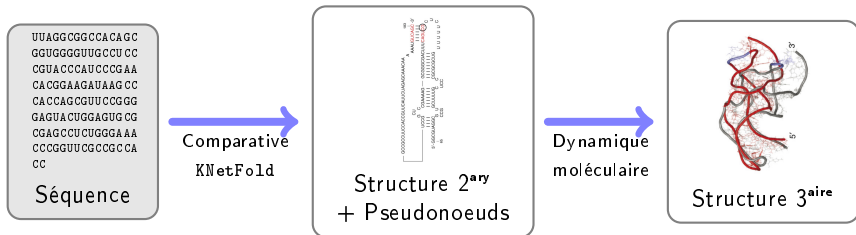
Séquence



Structure 3^aire

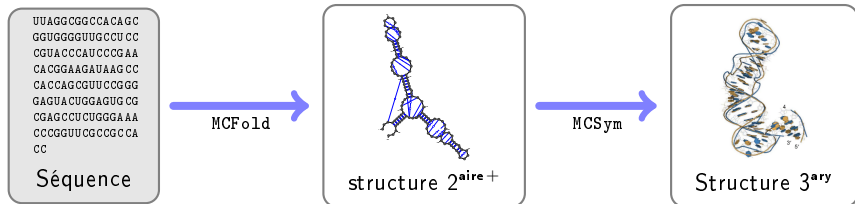
But : De la séquence à des modèles tri-dimensionnels!!!

- Modèles comparatifs + Dynamique moléculaires : RNA2D3D [SYKB07]



But : De la séquence à des modèles tri-dimensionnels!!!

- Pipeline MC-Fold/MC-sym [PM08]





A. Condon, B. Davy, B. Rastegari, S. Zhao, and F. Tarrant.

Classifying RNA pseudoknotted structures.
Theoretical Computer Science, 320(1) :35–50, 2004.



K. Doshi, J. J. Cannone, C. Cobaugh, and R. R. Gutell.

Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for rna secondary structure prediction.
BMC Bioinformatics, 5(1) :105, 2004.



P. Gardner and R. Giegerich.

A comprehensive comparison of comparative rna structure prediction approaches.
BMC Bioinformatics, 5(1) :140, 2004.



R. B. Lyngsø and C. N. S. Pedersen.

RNA pseudoknot prediction in energy-based models.
Journal of Computational Biology, 7(3-4) :409–427, 2000.



N. Leontis and E. Westhof.

Geometric nomenclature and classification of RNA base pairs.
RNA, 7 :499–512, 2001.



D.H. Mathews, J. Sabina, M. Zuker, and D.H. Turner.

Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure.
J Mol Biol, 288 :911–940, 1999.



Ján Maňuch, Chris Thachuk, Ladislav Stacho, and Anne Condon.

Np-completeness of the direct energy barrier problem without pseudoknots.
pages 106–115, 2009.



N. R. Markham and M. Zuker.

Bioinformatics, chapter UNAFold, pages 3–31.
Springer, 2008.



M. Parisien and F. Major.

The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data.

Nature, 452(7183) :51–55, 2008.



Lioudmila V Sharova, Alexei A Sharov, Timur Nedorezov, Yulan Piao, Nabeebi Shaik, and Minoru S H Ko.

Database for mrna half-life of 19 977 genes obtained by dna microarray analysis of pluripotent and differentiating mouse embryonic stem cells.

DNA Res, 16(1) :45–58, Feb 2009.



B. A. Shapiro, Y. G. Yingling, W. Kasprzak, and E. Bindewald.

Bridging the gap in rna structure prediction.

Curr Opin Struct Biol, 17(2) :157–165, Apr 2007.

Exercice : Parsing/repliement des structures secondaires (Python)

[http://www.lix.polytechnique.fr/~ponty/enseignement/
2012-01-BIBS-TP1-RappelsPython.pdf](http://www.lix.polytechnique.fr/~ponty/enseignement/2012-01-BIBS-TP1-RappelsPython.pdf)