M2 BIM - Lecture 3
Advanced dynamic programming and alignment

Yann Ponty

Bioinformatics Team
École Polytechnique/CNRS/INRIA AMIB – France

December 15th, 2015

Plan

1 **Quelques remarques et résultats techniques**
- Validité d'un schéma
- Structures sous-optimales
- Pseudo-noeuds

2 **Alignement et comparaison de structures d'ARN**
- Méthode géométrique
- Alignement de structures secondaires
- Méthodes hybrides

3 **Programmation dynamique générique**
- Introduction
- Analyse syntaxique
- Hypergraphes

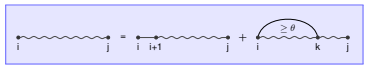
Validité d'un schéma

Une preuve de correction possible :

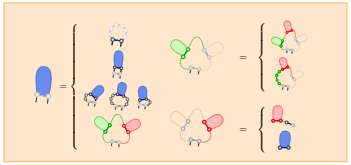
Calcul correct localement

+ Toutes les conformations sont parcourues

⇒ Algorithme correct (Induction)



$\overset{?}{\Leftrightarrow}$



Forte certitude **mais** pas encore preuve (Séries génératrices).

Validité d'un schéma

Une preuve de correction possible :

Calcul correct localement

+ Toutes les conformations sont parcourues

⇒ Algorithme correct (Induction)

$$C_{i,t} = 1, \quad \forall t \in [i, i+\theta]$$

$$C_{i,j} = \sum \begin{cases} C_{i+1,j} \\ \sum_{k=i+\theta+1}^{j} 1 \times C_{i+1,k-1} \times C_{k+1,j} \end{cases}$$

Homopolymère (Toute paire autorisée) + $\theta = 1$
$\Rightarrow C_{1,n} = \mathbf{1}, 1, 1, 2, 4, 8, 17, 32, 82, 185, 423, \ldots$

$\overset{?}{\Leftrightarrow}$

$$\mathcal{C}'_{i,j} = \sum \begin{cases} 1 \\ \mathcal{C}'_{i+1,j-1} \\ \sum_{i',j'} \mathcal{C}'_{i',j'} \\ \sum_k C_{i+1,k-1} \times \mathcal{C}^1{}_{k,j-1} \end{cases}$$

$$C_{i,j} = \sum_k \left((C_{i,k-1} + 1) \times \mathcal{C}^1{}_{k,j} \right)$$

$$\mathcal{C}^1{}_{i,j} = \mathcal{C}^1{}_{i,j-1} + \mathcal{C}'_{i,j}$$

Homopolymère + $\theta = 1$
$\Rightarrow \mathcal{C}'_{1,n} = \mathbf{0}, 1, 1, 2, 4, 8, 17, 32, 82, 185, 423, \ldots$

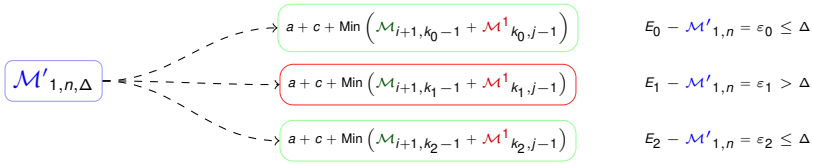
Forte certitude **mais** pas encore preuve (Séries génératrices).

Repliement sous-optimal

Prob. : Simplifications de l'énergie (Pseudo-noeuds, non-can.)
$\Rightarrow$ La structure **native** (fonctionnelle) pourrait être **ignorée**.

$\Rightarrow$ Engendrer des repliements sous-optimaux (`RNASubopt` [WFHS99]),
i.e. construire toutes les structures à Δ KCal.mol^{-1} de la MFE :

- Calculer la matrice des énergies minimales
- Effectuer un Backtrack sur toutes les contributions à $\leq \Delta$ de la MFE
- Mettre à jour Δ t.q. les futurs backtracks donnent ≥ 1 struct.
- Engendrer (Rec.) les sous-ensembles et combiner (brutal ou Tri)

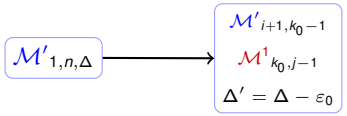


Repliement sous-optimal

Prob. : Simplifications de l'énergie (Pseudo-noeuds, non-can.)
$\Rightarrow$ La structure **native** (fonctionnelle) pourrait être **ignorée**.

$\Rightarrow$ Engendrer des repliements sous-optimaux (`RNASubopt` [WFHS99]),
i.e. construire toutes les structures à Δ KCal.mol^{-1} de la MFE :

- Calculer la matrice des énergies minimales
- Effectuer un Backtrack sur toutes les contributions à $\leq \Delta$ de la MFE
- Mettre à jour Δ t.q. les futurs backtracks donnent ≥ 1 struct.
- Engendrer (Rec.) les sous-ensembles et combiner (brutal ou Tri)

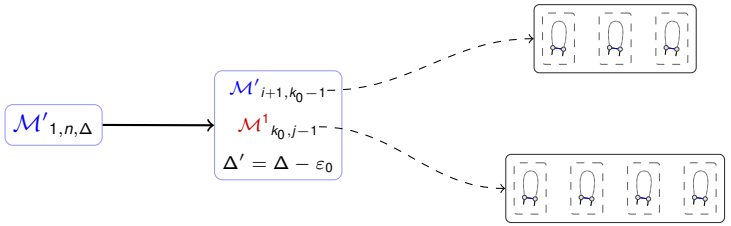


Repliement sous-optimal

Prob. : Simplifications de l'énergie (Pseudo-noeuds, non-can.)
$\Rightarrow$ La structure **native** (fonctionnelle) pourrait être **ignorée**.

$\Rightarrow$ Engendrer des repliements sous-optimaux (`RNASubopt` [WFHS99]),
i.e. construire toutes les structures à Δ KCal.mol^{-1} de la MFE :

- Calculer la matrice des énergies minimales
- Effectuer un Backtrack sur toutes les contributions à $\leq \Delta$ de la MFE
- Mettre à jour Δ t.q. les futurs backtracks donnent ≥ 1 struct.
- Engendrer (Rec.) les sous-ensembles et combiner (brutal ou Tri)

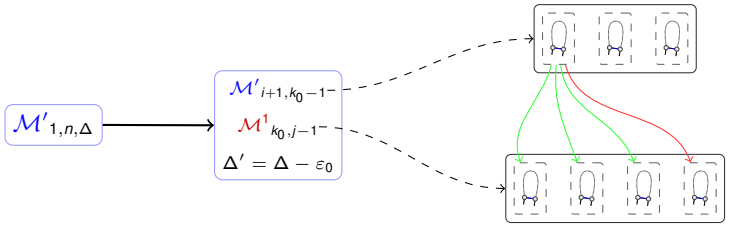


Repliement sous-optimal

Prob. : Simplifications de l'énergie (Pseudo-noeuds, non-can.)
$\Rightarrow$ La structure **native** (fonctionnelle) pourrait être **ignorée**.

$\Rightarrow$ Engendrer des repliements sous-optimaux (`RNASubopt` [WFHS99]),
i.e. construire toutes les structures à Δ KCal.mol^{-1} de la MFE :

- Calculer la matrice des énergies minimales
- Effectuer un Backtrack sur toutes les contributions à $\leq \Delta$ de la MFE
- Mettre à jour Δ t.q. les futurs backtracks donnent ≥ 1 struct.
- Engendrer (Rec.) les sous-ensembles et combiner (brutal ou Tri)

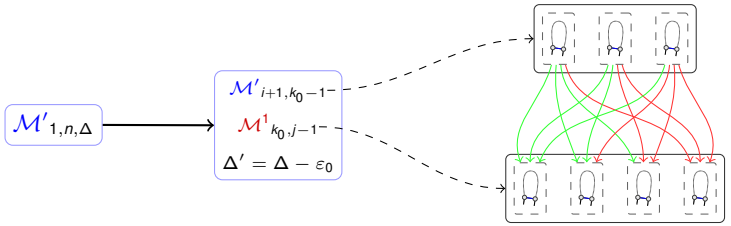


Repliement sous-optimal

Prob. : Simplifications de l'énergie (Pseudo-noeuds, non-can.)
$\Rightarrow$ La structure **native** (fonctionnelle) pourrait être **ignorée**.

$\Rightarrow$ Engendrer des repliements sous-optimaux (`RNASubopt` [WFHS99]),
i.e. construire toutes les structures à Δ KCal.mol^{-1} de la MFE :

- Calculer la matrice des énergies minimales
- Effectuer un Backtrack sur toutes les contributions à $\leq \Delta$ de la MFE
- Mettre à jour Δ t.q. les futurs backtracks donnent ≥ 1 struct.
- Engendrer (Rec.) les sous-ensembles et combiner (brutal ou Tri)



Repliement sous-optimal

Prob. : Simplifications de l'énergie (Pseudo-noeuds, non-can.)
$\Rightarrow$ La structure **native** (fonctionnelle) pourrait être **ignorée**.

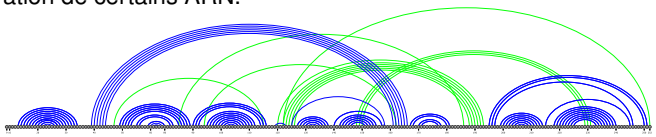
$\Rightarrow$ Engendrer des repliements sous-optimaux (`RNASubopt` [WFHS99]),
i.e. construire toutes les structures à Δ KCal.mol^{-1} de la MFE :

- Calculer la matrice des énergies minimales
- Effectuer un Backtrack sur toutes les contributions à $\leq \Delta$ de la MFE
- Mettre à jour Δ t.q. les futurs backtracks donnent ≥ 1 struct.
- Engendrer (Rec.) les sous-ensembles et combiner (brutal ou Tri)

$\Rightarrow$ Complexité en temps (Tri) : $\mathcal{O}(n^3 + nk\log(k))$
(k croît exponentiellement sur Δ !)

Repliement avec pseudo-noeuds

Les pseudo-noeuds (et vrais noeuds) sont des constituants essentiels à la structuration de certains ARN.



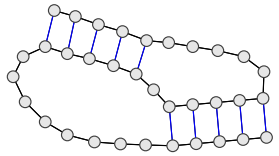
Ribozyme du groupe I

Leur absence historique au sein algorithmes de repliement est liée à la difficulté algorithmique des problèmes associés.
(Présence de croisements interdit une hypothèse d'indépendance des repliements).

Type	Complexité	Référence
Structures secondaires	$\mathcal{O}(n^3)$	[MSZT99]
L&P	$\mathcal{O}(n^5)$	[LP00]
D&P	$\mathcal{O}(n^5)$	[DP03]
A&U	$\mathcal{O}(n^5)$	[Aku00]
R&E	$\mathcal{O}(n^6)$	[RE99]
Généraux	NP-complet	[LP00]

Algorithme d'Akutsu/Uemura

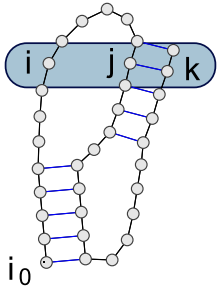
But : Capturer des catégorie de pseudo-noeuds *simples*, mais très représentés.



Idée : Quand on *retourne* ce type de pseudonoeuds, il suffit de précalculer les meilleures configurations *en dessous* d'un triplet (i, j, k) pour obtenir son énergie minimale.

Algorithme d'Akutsu/Uemura

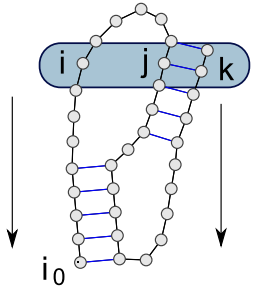
But : Capturer des catégorie de pseudo-noeuds *simples*, mais très représentés.



Idée : Quand on *retourne* ce type de pseudonoeuds, il suffit de précalculer les meilleures configurations *en dessous* d'un triplet (i, j, k) pour obtenir son énergie minimale.

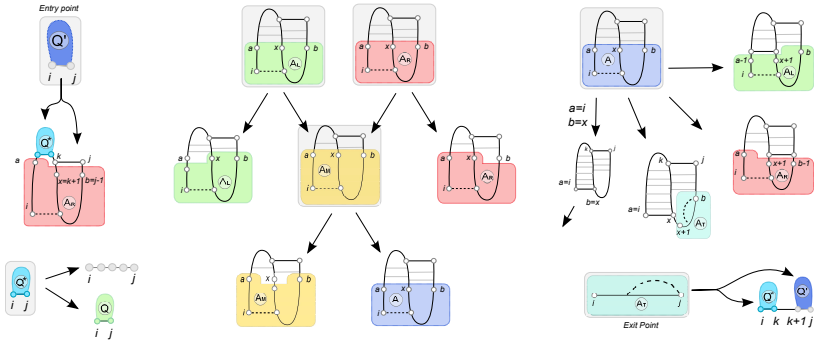
Algorithme d'Akutsu/Uemura

But : Capturer des catégorie de pseudo-noeuds *simples*, mais très représentés.



Idée : Quand on *retourne* ce type de pseudonoeuds, il suffit de précalculer les meilleures configurations *en dessous* d'un triplet (i, j, k) pour obtenir son énergie minimale.

Akutsu/Uemura : Programmation dynamique



Application/Problème	Weight fun.	Time/Space	Ref.
Minimisation d'énergie	π_{bp}	$\mathcal{O}(n^4)/\mathcal{O}(n^4)$	[Aku00]
Fonction de partition	$e^{\frac{-\pi_{bp}}{RT}}$	$\mathcal{O}(n^4)/\mathcal{O}(n^4)$	$\Theta(n^6)$[CC09]
Probabilité de paires de bases	$e^{\frac{-\pi_{bp}}{RT}}$	$\mathcal{O}(n^4)/\mathcal{O}(n^4)$	–
Échantillonnage (k-struct.)	$e^{\frac{-\pi_{bp}}{RT}}$	$\mathcal{O}(n^4 + kn\log n)/\mathcal{O}(n^4)$	–

Exercice : Ecrire l'équation de programmation dynamique associée pour le repliement, le comptage et la fonction de partition.

Plan

1 Quelques remarques et résultats techniques
- Validité d'un schéma
- Structures sous-optimales
- Pseudo-noeuds

2 Alignement et comparaison de structures d'ARN
- Méthode géométrique
- Alignement de structures secondaires
- Méthodes hybrides

3 Programmation dynamique générique
- Introduction
- Analyse syntaxique
- Hypergraphes

Pourquoi aligner structurellement des ARN

Hypothèse : Pression évolutive commune = fonction commune.

Chez certains organismes/familles d'ARN (ex : RNAse-P), très faible conservation de la séquence, mais structure reste conservée, et peut être déterminée expérimentalement (*in vitro* ou *in silico*).

Problèmes :

- **Édition :** Trouver la *distance* entre deux structures A et B. Quelle séquence d'opérations (de coût minimal) pour changer A en B ? Déjà NP-complet pour deux structures secondaires [BFRS07].
- **Alignement :** Trouver une super-structure de coût minimal. Généralise la notion d'alignement de séquence. Polynomial pour des structures secondaires [BDD+08], NP-complet en 3D [SZS+08]. **Variantes :** Alignement local ou global, Recherche de motifs.
- **Superposition :** Trouver une transformation géométrique (Rotation, translation, zoom) pour superposer *au mieux* les coordonnées de deux ARN de matching connu. Polynomial en 3D [McL82].

Remarque : Difficulté algorithme provient de la recherche d'un matching.

FR3D : Un exemple d'approche géométrique

Quand les structures tertiaires (3D) des ARN sont disponibles, le problème de l'alignement peut être abordé de façon purement géométrique.

Problème

Donnée : Motif m et structure cible b (Ensembles de bases 3D).
Résultat : Matching de m et d'un sous-ensemble de b minimisant une divergence géométrique.

Divergence géométrique : Dans FR3D [SZS+08], une fonction D basée sur deux fonctions L et A d'erreur tenant compte respectivement de la superposabilité (L) et de l'orientation des bases (A) de m et b.

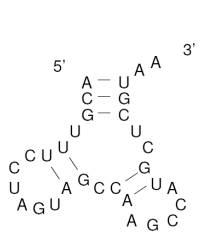
$$L = \sqrt{\min_{R,T} \sum_{i=1}^{m} \|b_i - R(T(m_i))\|^2} \quad A = \sqrt{\sum_{i=1}^{m} \alpha_i^2} \quad D = \frac{1}{m}\sqrt{L^2 + A^2}$$

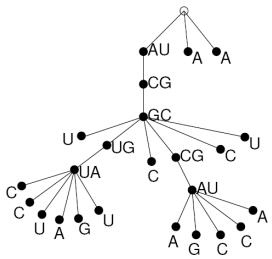
R, T : Rotation et translation. c_i : Barycentre de la base m_i. α_i : Écart entre les axes barycentre/bases dans m_i et b_i.

Exploration (Backtrack) + Élagage incrémental (Bornes sur D) $\Rightarrow$ Explosion !
Mais recherche exacte réalisable pour des petits motifs.

Structures vers arbres

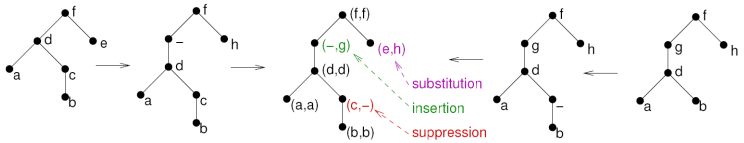
L'alignement de deux structures secondaires est basé sur une représentation arborescente de la structure secondaire [1].





Paires de bases $\Rightarrow$ noeuds internes Bases non-appariées $\Rightarrow$ Feuilles

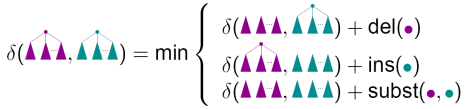
Alignement = Construction d'un matching complet de coût minimal.



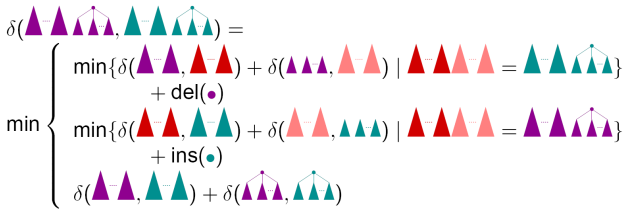
1. Illustrations empruntées à C. Herrbach

Alignement historique : Jiang, Wang & Zhang 95 [JWZ94]

Alignement d'arbre [2]



Alignement de forêt



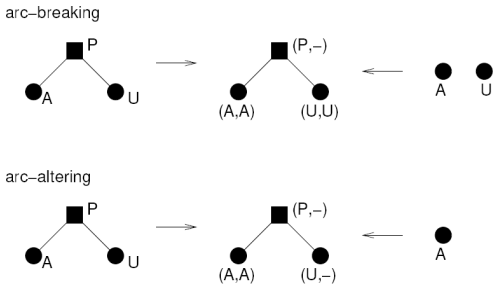
Complexité au pire en $\mathcal{O}(n^4)$ [JWZ94], en moyenne en $\mathcal{O}(n^2)$ [HDD07].
Mais opérations spécifiques à l'ARN manquantes.

2. Idem

RNAForester [HVG04]

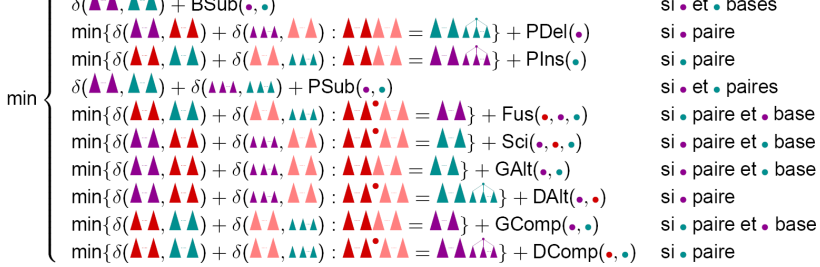
Possibilité de paramétrer les coûts des opérations, mais certaines opérations, atomiques dans un modèle réaliste, devront être recomposées à partir des opérations disponibles.
Exemple : Désappariement d'une paire de base nécessite une suppression (paire de base) et deux insertions (bases).

RNAForester : Basé sur l'algorithme de Jiang, Wang & Zhang
+ Intégrations d'opérations spécifiques à l'ARN[3].



3. Idem

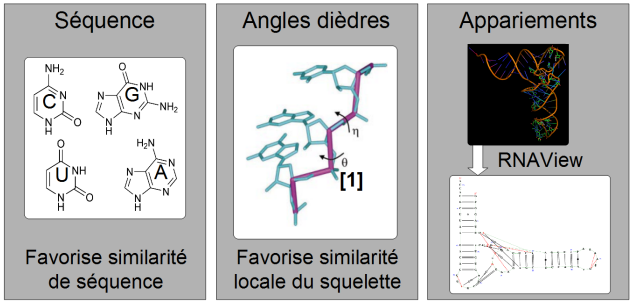
NestedAlign [BDD+08]



Méthode hybride

DIAL [FPLC07] est une méthode hybride qui se concentre sur les comportements locaux.
Idée : L'ARN est flexible, petite variation locale peuvent entraîner des grandes déviations géométriques.

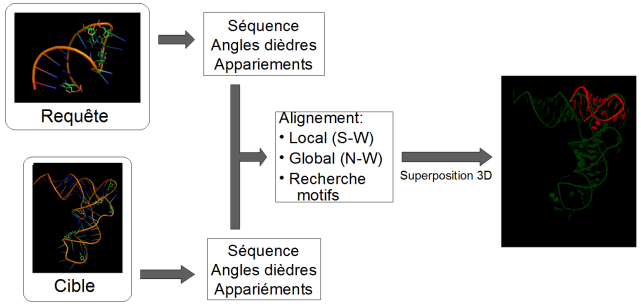
DIAL capture les similarités locales à trois niveau :



Méthode hybride

DIAL [FPLC07] est une méthode hybride qui se concentre sur les comportements locaux.
Idée : L'ARN est flexible, petite variation locale peuvent entraîner des grandes déviations géométriques.

Un algorithme d'alignement de séquence est alors utilisé



Conclusion

Tout dépend de ce que l'on a et veut :

- Modèle 3D :
  - Recherche d'un motif peu conservé en séquence : FR3D
  - Recherche d'un motif conservé : FR3D, DIAL ou DARTS
  - Recherche d'une structure entière : DIAL ou DARTS
- Structure secondaire :
  - Recherche d'un motif : NestedAlign
  - Alignement structure : RNAForester, NestedAlign

De nombreux autres programmes disponibles : Migal, Magnolia, . . .
+ Explosion des approches *par fragments* : FASTR3D, RNA FRABASE, . . .

Plan

1 Quelques remarques et résultats techniques
- Validité d'un schéma
- Structures sous-optimales
- Pseudo-noeuds

2 Alignement et comparaison de structures d'ARN
- Méthode géométrique
- Alignement de structures secondaires
- Méthodes hybrides

3 **Programmation dynamique générique**
- Introduction
- Analyse syntaxique
- Hypergraphes

Rappels

> *The development of successful dynamic programming recurrences is a matter of experience, talent and luck. (Relecteur anonyme, 2000)*

Tout algorithme de programmation dynamique définit :

1. Espace de recherche
2. Fonction objectif ou score
3. **Décomposition**

Entités conceptuelles non-indépendantes :

- **Décomposition** doit parcourir tout l'Espace de recherche de façon unique
- **Décomposition** doit permettre un calcul **local** de la Fonction objectif

Choix d'une bonne décomposition est crucial !

Compilation de ces trois éléments ⇒ **Équation de programmation dynamique**.
Question : Comment **formaliser** cette notion de compilation ?

Deux grandes familles de formalismes

- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).
- Approche Hypergraphe :
  Espace de recherche = (hyper-)chemins d'un hypergraphe.

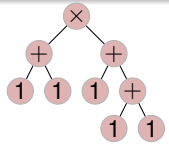
Deux grandes familles de formalismes

- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).

  Ex1 (Caravane d'El Mamoun) :
  Grammaire $= \{S \to S + S \mid S \times S \mid 1\}$
  Quelle(s) valeur(s) pour une expression arithmétique sans parenthésage :
  $$1 + 1 \times 1 + 1 + 1 = \ ?$$
  En général : Se demander comment parenthéser l'expression de façon à min (resp.max)-imiser la valeur de l'expression est un problème d'analyse syntaxique (pondéré).

- Approche Hypergraphe :
  Espace de recherche = (hyper-)chemins d'un hypergraphe.

Deux grandes familles de formalismes

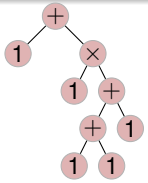


- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).

  Ex1 (Caravane d'El Mamoun) :
  Grammaire $= \{S \to S + S \mid S \times S \mid 1\}$
  Quelle(s) valeur(s) pour une expression arithmétique sans parenthésage :
  $$1 + 1 \times 1 + 1 + 1 = \ = (1 + 1) \times (1 + (1 + 1)) = 6\,?$$
  En général : Se demander comment parenthéser l'expression de façon à min (resp.max)-imiser la valeur de l'expression est un problème d'analyse syntaxique (pondéré).

- Approche Hypergraphe :
  Espace de recherche = (hyper-)chemins d'un hypergraphe.

Deux grandes familles de formalismes

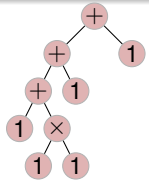


- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).

  Ex1 (Caravane d'El Mamoun) :
  Grammaire $= \{S \to S + S \mid S \times S \mid 1\}$
  Quelle(s) valeur(s) pour une expression arithmétique sans parenthésage :
  $$1 + 1 \times 1 + 1 + 1 = \ = 1 + (1 \times ((1 + 1) + 1)) = 4\,?$$
  En général : Se demander comment parenthéser l'expression de façon à min (resp.max)-imiser la valeur de l'expression est un problème d'analyse syntaxique (pondéré).

- Approche Hypergraphe :
  Espace de recherche = (hyper-)chemins d'un hypergraphe.

Deux grandes familles de formalismes



- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).

  Ex1 (Caravane d'El Mamoun) :
  Grammaire $= \{S \to S + S \mid S \times S \mid 1\}$
  Quelle(s) valeur(s) pour une expression arithmétique sans parenthésage :
  $$1 + 1 \times 1 + 1 + 1 = \ = ((1 + (1 \times 1)) + 1) + 1 = 4\,?$$
  En général : Se demander comment parenthéser l'expression de façon à min (resp.max)-imiser la valeur de l'expression est un problème d'analyse syntaxique (pondéré).

- Approche Hypergraphe :
  Espace de recherche = (hyper-)chemins d'un hypergraphe.

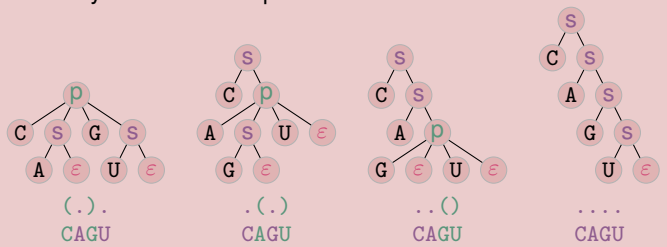
Deux grandes familles de formalismes

- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).

Ex2 (ARN) :

$R \rightarrow \mathtt{A}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{A}\,R \mid \mathtt{G}\,R\,\mathtt{C}\,R \mid \mathtt{C}\,R\,\mathtt{G}\,R \mid \mathtt{G}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{G}\,R$ (Règles p)
$\quad \mid \mathtt{A}\,R \mid \mathtt{C}\,R \mid \mathtt{G}\,R \mid \mathtt{U}\,R \mid \varepsilon$ (Règles s)

Arbres de syntaxe abstraite pour CAGU :



- Approche Hypergraphe :
  Espace de recherche = (hyper-)chemins d'un hypergraphe.

Deux grandes familles de formalismes

- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).

Ex2 (ARN) :

$R \rightarrow \mathtt{A}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{A}\,R \mid \mathtt{G}\,R\,\mathtt{C}\,R \mid \mathtt{C}\,R\,\mathtt{G}\,R \mid \mathtt{G}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{G}\,R$ (Règles p)
$\quad \mid \mathtt{A}\,R \mid \mathtt{C}\,R \mid \mathtt{G}\,R \mid \mathtt{U}\,R \mid \varepsilon$ (Règles s)

+ Pondération des règles (Ex : 1/0 pour p/s)
$\Rightarrow$ Repliement à la Nussinov = Parsing pondéré **sans équation explicite !**

Intérêt :
- Algorithmique générique en $O(n^3)$
- Pas de manipulation des indices $\Rightarrow$ moins d'erreurs ...
- Grammaires plus modulaires : Robustesse au changement de modèle
- Séparation assez naturelle espace de recherche/score d'évaluation
- Extensions possibles à certaines grammaires faiblement contextuelles (surcoût algorithmique)

Inconvénients :
- Apprendre le formalisme
- Performances : Pertes de constantes d'implémentation, voir d'ordre de grandeur algorithmique pour structures plus complexes
- Problèmes d'ambiguités *sémantique* déjà complexes à définir

Approche Hypergraphe :
Espace de recherche = (hyper-)chemins d'un hypergraphe.

Deux grandes familles de formalismes

- Approches *analyse syntaxique* (parsing) :
  Modèle : Grammaire non-contextuelle
  Espace de recherche = Arbres de syntaxe abstraite (parse trees).

Ex2 (ARN) :

$R \rightarrow \mathtt{A}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{A}\,R \mid \mathtt{G}\,R\,\mathtt{C}\,R \mid \mathtt{C}\,R\,\mathtt{G}\,R \mid \mathtt{G}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{G}\,R$ (Règles p)
$\quad \mid \mathtt{A}\,R \mid \mathtt{C}\,R \mid \mathtt{G}\,R \mid \mathtt{U}\,R \mid \varepsilon$ (Règles s)

+ Pondération des règles (Ex : 1/0 pour p/s)
$\Rightarrow$ Repliement à la Nussinov = Parsing pondéré **sans équation explicite !**

Intérêt :
- Algorithmique générique en $O(n^3)$
- Pas de manipulation des indices $\Rightarrow$ moins d'erreurs ...
- Grammaires plus modulaires : Robustesse au changement de modèle
- Séparation assez naturelle espace de recherche/score d'évaluation
- Extensions possibles à certaines grammaires faiblement contextuelles (surcoût algorithmique)

Inconvénients :
- Apprendre le formalisme
- Performances : Pertes de constantes d'implémentation, voir d'ordre de grandeur algorithmique pour structures plus complexes
- Problèmes d'ambiguités *sémantique* déjà complexes à définir

Approche Hypergraphe :
Espace de recherche = (hyper-)chemins d'un hypergraphe.

Deux implémentations de l'analyses syntaxique générique

$$
\left.\begin{array}{l}
V_T = \{0, \cdots, 9, +, \times\}, \\
V_N = S, \\
P = \begin{cases} S \rightarrow S + S \\ S \rightarrow S \times S \\ S \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{cases}
\end{array}\right\} \text{Context-free grammar}
$$

$$
\left.\begin{array}{l}
\mathcal{A} = \mathbb{N} \\
\lambda_{\mathcal{A}} = \begin{cases} S_{\mathcal{A}}(0) = 0 \\ \vdots \\ S_{\mathcal{A}}(9) = 9 \\ S_{\mathcal{A}}(+, \times) = 0 \end{cases} \\
F_P = \begin{cases} f_{S \rightarrow S+S}(xyz) = x + z \\ f_{S \rightarrow S \times S}(xyz) = x \times z \\ f_{S \rightarrow a \in V_T}(x) = x \end{cases}
\end{array}\right\} \text{Attribute system}
$$

[Crédit : J. Waldispühl]

- **Grammaires (multibandes) attribuées (Lefebvre, Waldispühl *et al*)**
  Calculer/minimiser un score sur un système d'attribut.
  Avantages : Pseudonoeuds simples peuvent être explorés grace à des grammaires multibandes en *synchronisant* les analyses syntaxiques .
  Inconvénient : Induction de surcoût algorithmique substantiel pour certaines familles *légèrement* contextuelles (**ex :** Repliement avec pseudonoeuds).

`http://people.csail.mit.edu/jw/software.php`

Deux implémentations de l'analyses syntaxique générique

Grammaire d'analyse (yield grammar) :

```
>nussinov78 alg inp = axiom s where
> (nil,left,right,pair,split,h) = alg
> s = tabulated (
>   nil   <<< empty                 |||
>   left  <<< base -~~ s            |||
>   right <<<          s ~~- base   |||
>   (pair <<< base -~~ s ~~- base)
>               `with` basepairing |||
>   split <<<       s +~+ s        ... h)
```

Algèbre d'évaluation (base-pair algebra) :

```
>pairmax :: Nussinov_Algebra Char Int
>pairmax = (nil,left,right,pair,
            split,h) where
>  nil _          = 0
>  left  _ x      = x
>  right   x _    = x
>  pair  _ x _    = x + 1
>  split x y      = x + y
>  h xs           = [maximum xs]
```

- **Programmation dynamique algébrique (ADP, Giegerich *et al*)**
  Avantages :
  - Applications plus générale grâce à une séparation de grammaires respectivement consacrées à l'analyse syntaxique et à l'évaluation.
  - Cadre formel et implémentation *in-extenso* (> 20 publis, dont 3 thèses).

  Inconvénient :
  - Notations cryptiques, implémentation liée fortement au langage Haskell.
  - Pseudonoeuds : Formalisme doit être *hacké*, perdant une partie des bénéfice de l'approche générique.
  - Structure combinatoire disparaît derrière des notations (trop ?) abstraites.
    ⇒ Gros problèmes d'ambiguité, impossible à traiter de façon purement automatisée (Problèmes *indécidables*).

  `http://bibiserv.techfak.uni-bielefeld.de/adp/`

Trois grandes familles de formalismes

Hypergraphes (Roytberg/Finkelstein,...)
Avantages :
- Très grande expressivité.
- Toute grammaire non-contextuelle peut être transformée en un hypergraphe équivalent (Hyperchemins en bijection avec les parse trees).
- Influence limitée de l'ordre.

Inconvénient :
- Pas (encore) d'implémentation générique.
- Travail de modélisation plus conséquent.
  En particulier, la question de l'ambiguité est laissée à la charge du concepteur, mais les conditions d'applications des algorithmes génériques correspondent à des problèmes décidables.
- Manipulation explicite des indices (Force et faiblesse).

Pourquoi les hypergraphes ?

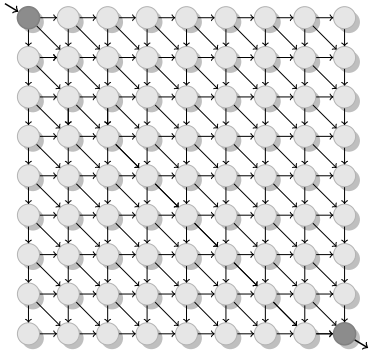
Rappel : Dans **Needleman-Wunsch (alignement global)**, la remontée (ou backtrack) correspond à un chemin de coût minimal dans la matrice, ie un chemin dont la somme des coûts des arêtes est minimale.

	A	C	A	C	A	C	T	A
	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

Remarque : L'absence de cycle facilite ici la programmation dynamique.
Malheureusement, on ne peut pas prolonger cette analogie vers Nussinov car les schémas de backtracks sont analogues à des arbres.
⇒ Il faudrait des graphes dont les chemins sont des arbres → Hypergraphes !

Pourquoi les hypergraphes ?

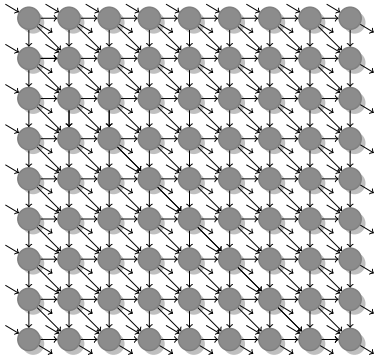
Rappel : Dans **Needleman-Wunsch (alignement global)**, la remontée (ou backtrack) correspond à un chemin de coût minimal dans la matrice, ie un chemin dont la somme des coûts des arêtes est minimale.



Remarque : L'absence de cycle facilite ici la programmation dynamique.
Malheureusement, on ne peut pas prolonger cette analogie vers Nussinov car les schémas de backtracks sont analogues à des arbres.
⇒ Il faudrait des graphes dont les chemins sont des arbres → Hypergraphes !

Pourquoi les hypergraphes ?

Rappel : Dans Smith-Waterman, la remontée (ou backtrack) correspond à un chemin de coût minimal dans la matrice, ie un chemin dont la somme des coûts des arêtes est minimale.

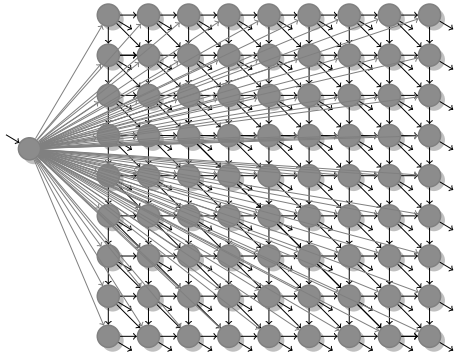


Remarque : L'absence de cycle facilite ici la programmation dynamique.

Malheureusement, on ne peut pas prolonger cette analogie vers Nussinov car les schémas de backtracks sont analogues à des arbres.
$\Rightarrow$ Il faudrait des graphes dont les chemins sont des arbres $\rightarrow$ Hypergraphes !

Pourquoi les hypergraphes ?

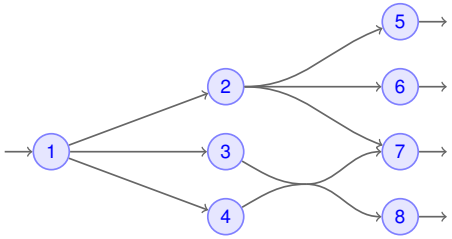
Rappel : Dans Smith-Waterman, la remontée (ou backtrack) correspond à un chemin de coût minimal dans la matrice, ie un chemin dont la somme des coûts des arêtes est minimale.



Remarque : L'absence de cycle facilite ici la programmation dynamique.

Malheureusement, on ne peut pas prolonger cette analogie vers Nussinov car les schémas de backtracks sont analogues à des arbres.
$\Rightarrow$ Il faudrait des graphes dont les chemins sont des arbres $\rightarrow$ Hypergraphes !

Hypergraphes



Les **hypergaphes** généralisent les graphes (dirigés) à des (hyper) arcs de degrés entrant/sortant arbitraires.

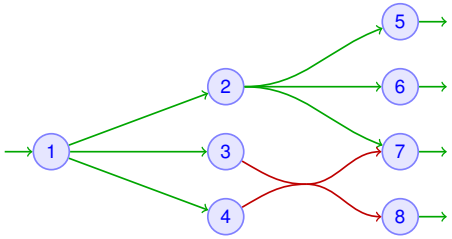
Definition (Hypergraphes)

Un **hypergraphe** $\mathcal{H}$ est un couple (V, E) tel que :

- V est un ensemble de sommet
- E est un ensemble d'hyperarcs $e = (t(e) \rightarrow h(e))$ tels que $t(e), h(e) \subset E$

Les hypergraphes avant, ou **F(orward)-graphes**, sont des hypergraphes dont les arcs ont degré entrant exactement égal à 1.

Hypergraphes



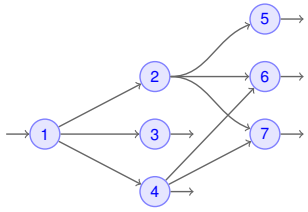
Les **hypergaphes** généralisent les graphes (dirigés) à des (hyper) arcs de degrés entrant/sortant arbitraires.

Definition (Hypergraphes)

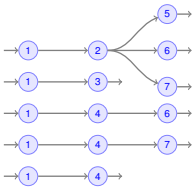
Un **hypergraphe** $\mathcal{H}$ est un couple (V, E) tel que :

- V est un ensemble de sommet
- E est un ensemble d'hyperarcs $e = (t(e) \rightarrow h(e))$ tels que $t(e), h(e) \subset E$

Les hypergraphes avant, ou **F(orward)-graphes**, sont des hypergraphes dont les arcs ont degré entrant exactement égal à 1.



F-graphe



Tous les F-chemins partant du sommet 1

Definition (F-chemin)

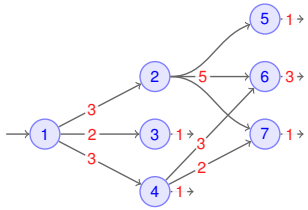
Un **F-chemin** est un arbre de racine $s \in V$, et dont les fils sont des F-chemins construits à partir des sommets sortants d'un arc $e = (s \to \mathbf{t}) \in E$.

Remarque : Les arcs ayant degré sortant 0 ($\mathbf{t} = \varnothing$) fournissent un cas terminal à cette définition recursive.

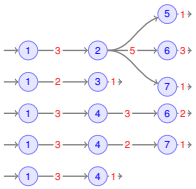
Un F-graphe est **indépendant** si et seulement si chacun de ses chemins emprunte au plus une fois chaque arc.

Soit $\pi : E \to \mathbb{R}$ un valuation **valuation**, associant une valeur numérique à chaque arc $e \in E$. On peut alors définir des notions de :

- **Poids** d'un chemin, égal au **produit** des valeurs de ses arcs.
- **Score** d'un chemin, égal au **la somme** des valeurs de ses arcs.



F-graphe



Tous les F-chemins partant du sommet 1

Definition (F-chemin)

Un **F-chemin** est un arbre de racine $s \in V$, et dont les fils sont des F-chemins construits à partir des sommets sortants d'un arc $e = (s \to \mathbf{t}) \in E$.

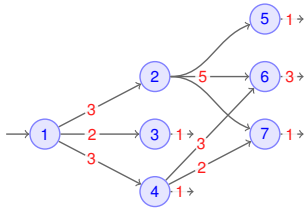
Remarque : Les arcs ayant degré sortant 0 ($\mathbf{t} = \varnothing$) fournissent un cas terminal à cette définition recursive.

Un F-graphe est **indépendant** si et seulement si chacun de ses chemins emprunte au plus une fois chaque arc.

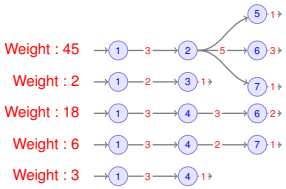
Soit $\pi : E \to \mathbb{R}$ un valuation **valuation**, associant une valeur numérique à chaque arc $e \in E$. On peut alors définir des notions de :

- **Poids** d'un chemin, égal au **produit** des valeurs de ses arcs.
- **Score** d'un chemin, égal au **la somme** des valeurs de ses arcs.

Yann Ponty M2 BIM - Lecture 3 - Advanced DP and alignment



F-graphe



Tous les F-chemins partant du sommet 1

Definition (F-chemin)

Un **F-chemin** est un arbre de racine $s \in V$, et dont les fils sont des F-chemins construits à partir des sommets sortants d'un arc $e = (s \to \mathbf{t}) \in E$.

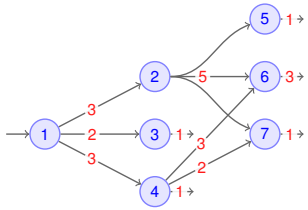
Remarque : Les arcs ayant degré sortant 0 ($\mathbf{t} = \varnothing$) fournissent un cas terminal à cette définition recursive.

Un F-graphe est **indépendant** si et seulement si chacun de ses chemins emprunte au plus une fois chaque arc.

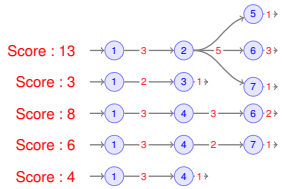
Soit $\pi : E \to \mathbb{R}$ un valuation **valuation**, associant une valeur numérique à chaque arc $e \in E$. On peut alors définir des notions de :

- **Poids** d'un chemin, égal au **produit** des valeurs de ses arcs.
- **Score** d'un chemin, égal au **la somme** des valeurs de ses arcs.

Yann Ponty M2 BIM - Lecture 3 - Advanced DP and alignment



F-graphe



Tous les F-chemins partant du sommet 1

Definition (F-chemin)

Un **F-chemin** est un arbre de racine $s \in V$, et dont les fils sont des F-chemins construits à partir des sommets sortants d'un arc $e = (s \to \mathbf{t}) \in E$.

Remarque : Les arcs ayant degré sortant 0 ($\mathbf{t} = \varnothing$) fournissent un cas terminal à cette définition recursive.

Un F-graphe est **indépendant** si et seulement si chacun de ses chemins emprunte au plus une fois chaque arc.

Soit $\pi : E \to \mathbb{R}$ un valuation **valuation**, associant une valeur numérique à chaque arc $e \in E$. On peut alors définir des notions de :

- **Poids** d'un chemin, égal au **produit** des valeurs de ses arcs.
- **Score** d'un chemin, égal au **la somme** des valeurs de ses arcs.

Yann Ponty M2 BIM - Lecture 3 - Advanced DP and alignment

Algorithmes basiques

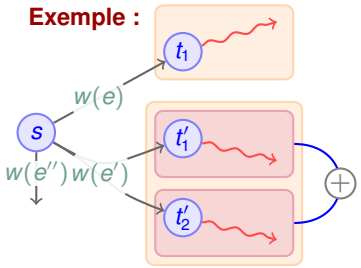
$\mathcal{H} = (s_0, V, E, \pi)$: hypergraphe acyclique s_0 : Sommet initial π : valuation

Quelques questions *naturelles* :

- Quel est le score (min/max)imal m_{s_0} d'un F-chemin partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.
- Quel est le nombre n_{s_0} de F-chemins partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.
- Quel poids total w_{s_0} de tous les F-chemins partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.

$$m_s = \min_{e=(s\to\mathbf{t})\in E} \left(w(e) + \sum_{u\in\mathbf{t}} m_u \right)$$

Exemple :



Algorithmes basiques

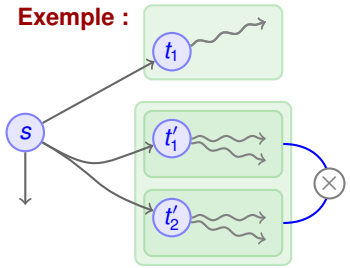
$\mathcal{H} = (s_0, V, E, \pi)$: hypergraphe acyclique s_0 : Sommet initial π : valuation

Quelques questions *naturelles* :

- Quel est le score (min/max)imal m_{s_0} d'un F-chemin partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.
- Quel est le nombre n_{s_0} de F-chemins partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.
- Quel poids total w_{s_0} de tous les F-chemins partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.

$$n_s = \sum_{(s\to\mathbf{t})\in E} \prod_{u\in\mathbf{t}} n_u$$

Exemple :



Algorithmes basiques

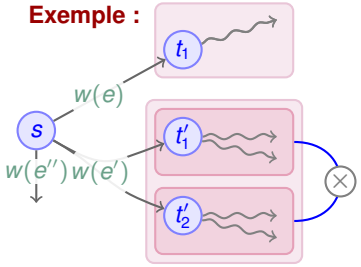
$\mathcal{H} = (s_0, V, E, \pi)$: hypergraphe acyclique s_0 : Sommet initial π : valuation

Quelques questions *naturelles* :

- Quel est le score (min/max)imal m_{s_0} d'un F-chemin partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.
- Quel est le nombre n_{s_0} de F-chemins partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.
- Quel poids total w_{s_0} de tous les F-chemins partant de $s_0 \in V$?
  $\Rightarrow$ **Complexités :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire.

$$w_s = \sum_{e=(s\to\mathbf{h}(e))\in E} w(e) \cdot \prod_{s'\in\mathbf{h}(e)} w_{s'}$$

Exemple :



Definition (Distribution pondérée)

Supposons une distribution pondérée sur l'ensemble $\mathcal{T}$ des F-chemins :

$$\mathbb{P}(p|\pi) = \frac{\prod_{e\in p} \pi(e)}{w_{s_0}}, \forall p \in \mathcal{T}.$$

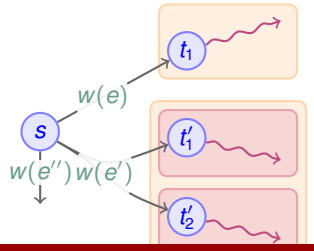
De nouvelles questions *ensemblistes* se posent alors :

- Comment engendrer $p \in \mathcal{T}$ aléatoirement selon la distribution pondérée ?
  $\Rightarrow$ **Complexité :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire **précalcul**
  + $\mathcal{O}(|p| + \sum_{e\in p} |\mathbf{h}(e)|)$ time **génération**.
- Quelle probabilité pour un arc donné d'être dans un F-chemin aléatoire ?
  $\Rightarrow$ Algorithme inside/outside
- Distribution(s) de valuation(s) additive(s) ...

Algorithme :
Choisir $e_i = (s \to \mathbf{t}_i)$ avec probabilité $p_{s,i}$ telle que :

$$p_{s,i} = \frac{w(e) \cdot \prod_{s'\in\mathbf{t}} w_{s'}}{w_s}$$

Réitérer récursivement sur tous les $\mathbf{t}_i$.



Definition (Distribution pondérée)

Supposons une distribution pondérée sur l'ensemble $\mathcal{T}$ des F-chemins :

$$\mathbb{P}(p|\pi) = \frac{\prod_{e \in p} \pi(e)}{w_{s_0}}, \forall p \in \mathcal{T}.$$

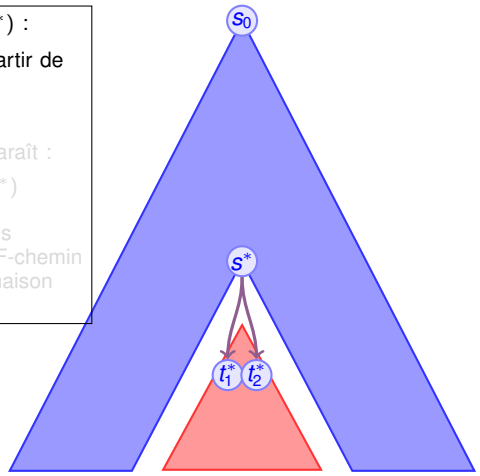
De nouvelles questions *ensemblistes* se posent alors :

- Comment engendrer $p \in \mathcal{T}$ aléatoirement selon la distribution pondérée ?
  $\Rightarrow$ **Complexité :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire **précalcul** + $\mathcal{O}(|p| + \sum_{e \in p} |\mathbf{h}(e)|)$ time **génération**.
- Quelle probabilité pour un arc donné d'être dans un F-chemin aléatoire ?
  $\Rightarrow$ Algorithme inside/outside
- Distribution(s) de valuation(s) additive(s) ...

Algorithme Inside/outside

F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

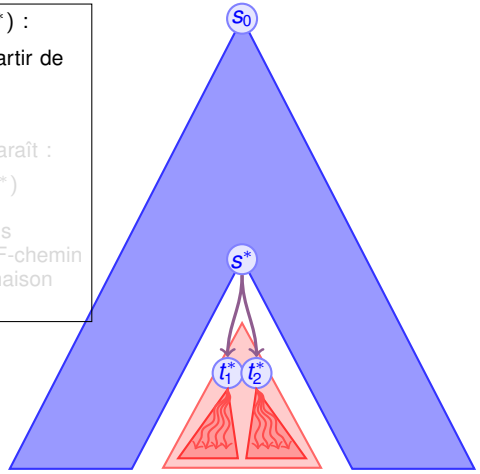
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : **Contexte** où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, **compléter** le F-chemin en explorant toute combinaison de frères pour p.



Algorithme Inside/outside

F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

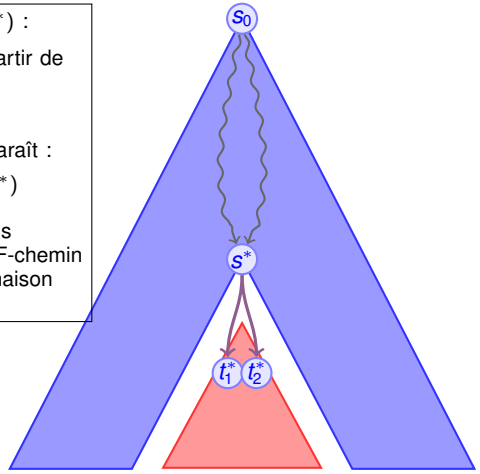
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : **Contexte** où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, **compléter** le F-chemin en explorant toute combinaison de frères pour p.



Algorithme Inside/outside

F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

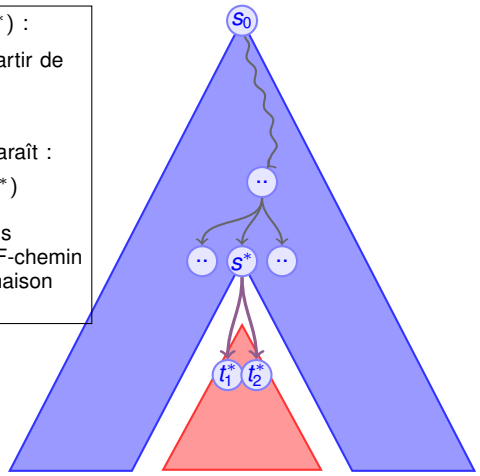
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : **Contexte** où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, **compléter** le F-chemin en explorant toute combinaison de frères pour p.



Algorithme Inside/outside

F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

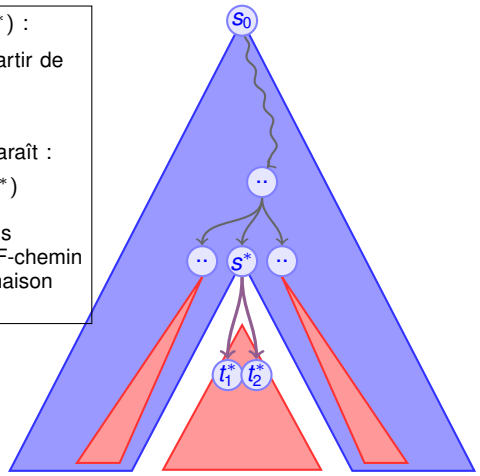
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : **Contexte** où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, **compléter** le F-chemin en explorant toute combinaison de frères pour p.



Algorithme Inside/outside

F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

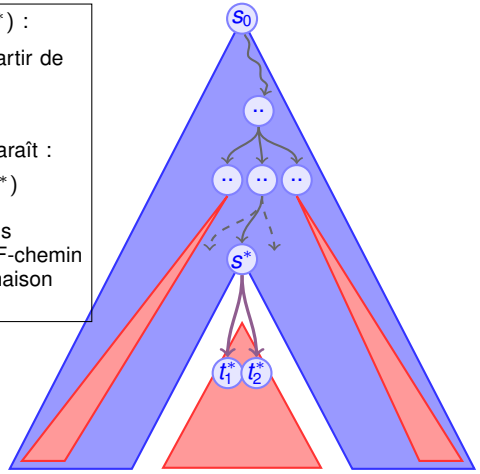
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : **Contexte** où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, **compléter** le F-chemin en explorant toute combinaison de frères pour p.



Algorithme Inside/outside

F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

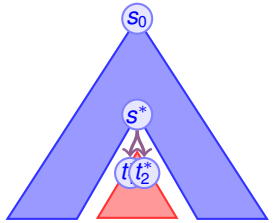
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : **Contexte** où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, **compléter** le F-chemin en explorant toute combinaison de frères pour p.



Algorithme Inside/outside

F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : **Contexte** où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, **compléter** le F-chemin en explorant toute combinaison de frères pour p.



Si le F-graphe est **acyclique** et **indépendant**, cette décomposition est **complète** et **non-ambiguë**, et implique le système suivant pour la probabilité cumulée p_{e^*} de tous les F-chemins empruntant $e^* = (s^* \to \mathbf{t}^*)$:

$$p_{e^*} = \frac{b_{s^*} \cdot \pi(e) \cdot \prod_{s' \in t^*} w_{s'}}{w_{s_0}}$$

$$b_s = \mathbf{1}_{s=s_0} + \sum_{\substack{e'=(s' \to \mathbf{t}') \in E \\ \text{s. t. } s \in \mathbf{t}}} \pi(e') \cdot b_{s'} \cdot \prod_{\substack{s'' \in \mathbf{t}' \\ s'' \neq s}} w_{s''}, \quad \forall s \in V$$

Applications ensemblistes

Definition (Distribution pondérée)

Supposons une distribution pondérée sur l'ensemble $\mathcal{T}$ des F-chemins :

$$\mathbb{P}(p|\pi) = \frac{\prod_{e \in p} \pi(e)}{w_{s_0}}, \forall p \in \mathcal{T}.$$

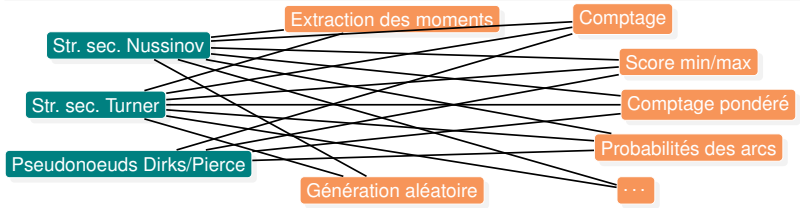
De nouvelles questions ensemblistes se posent alors :

- Comment engendrer $p \in \mathcal{T}$ aléatoirement selon la distribution pondérée ?
  $\Rightarrow$ **Complexité :** $\Theta(|E| + |V|)$ temps/$\Theta(|V|)$ mémoire **précalcul** + $\mathcal{O}(|p| + \sum_{e \in p} |\mathbf{h}(e)|)$ time **génération**.
- Quelle probabilité pour un arc donné d'être dans un F-chemin aléatoire ?
  $\Rightarrow$ Algorithme **inside/outside**
  $\Rightarrow$ **Complexité :** $\Theta(|E| + |V| + \sum_{e \in E} h(e)^2)$ temps/$\Theta(|V|)$ mémoire
- Distribution(s) de valuation(s) additive(s) (Moments d'une distribution pondérée) ...

Mi-temps

Message #1

Applications spécifiques de la programmation dynamique peuvent (et doivent) être détachées de l'équation, et être exprimée à un niveau abstrait.



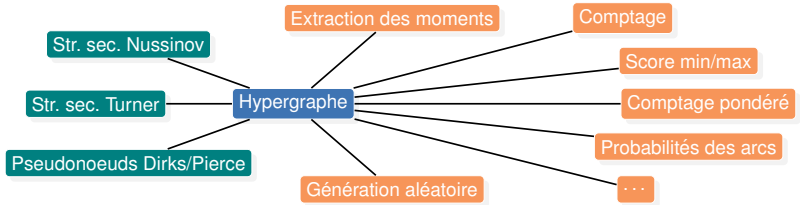
Credits : Roytberg and Finkelstein pour le *parallèle* Hypergraphe en Bioinformatique, L. Hwang pour une jolie formalisation algébriques de la programmation dynamique hypergraphe, Flajolet *et al* pour des transformations symboliques utilisé pour l'extraction des moments ...

Mais que faire de tout ça ?

Mi-temps

Message #1

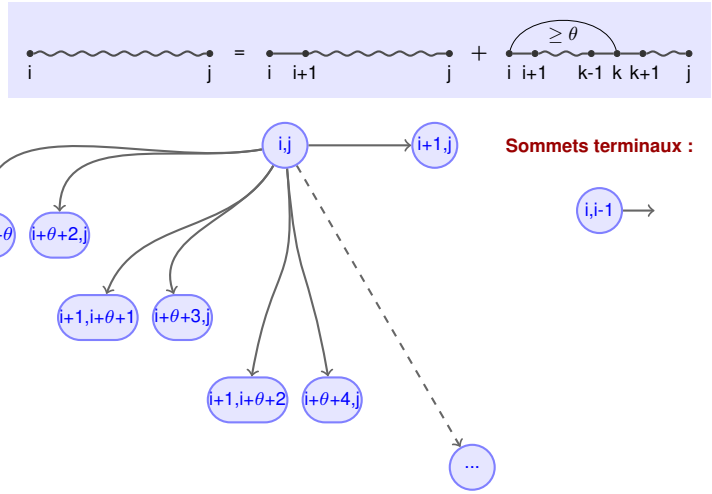
Applications spécifiques de la programmation dynamique peuvent (et doivent) être détachées de l'équation, et être exprimée à un niveau abstrait.



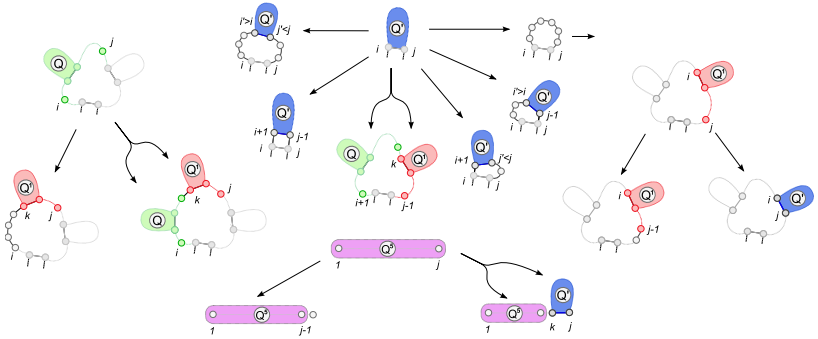
Credits : Roytberg and Finkelstein pour le *parallèle* Hypergraphe en Bioinformatique, L. Hwang pour une jolie formalisation algébriques de la programmation dynamique hypergraphe, Flajolet *et al* pour des transformations symboliques utilisé pour l'extraction des moments ...

Mais que faire de tout ça ?

Nussinov



Décomposition Mfold/Unafold



Cette décomposition est non-ambiguë et complète, et donne les arcs d'un F-graphe a $\Theta(n^2)$ sommets et $\mathcal{O}(n^3)$ qui est :

- Acyclique : Largeur d'intervalle strictement décroissante le long d'un arc.
- Independant : Intervalles disjoints deux-à-deux en sortie d'un arc.

Complexité du repliement en structure secondaire

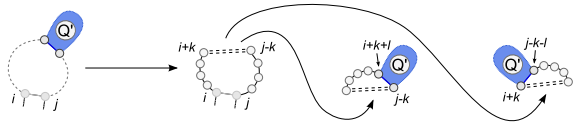


FIGURE – Stratégies alternatives pour les boucles internes, créant tout d'abord la partie symétrique en premier, puis la partie asymétrique.

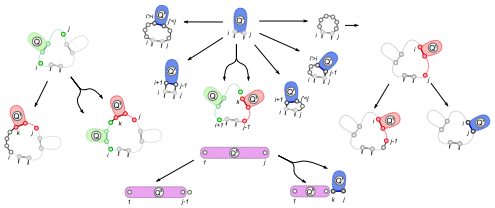
Du fait des boucles internes, l'ensemble des F-arcs engendrés pour le cas Q' est de cardinalité en $\mathcal{O}(n^4)$, ce qui contredit la complexité annoncée en $\mathcal{O}(n^3)$. Deux niveaux de réponses à cela :

- **Réponse 1 :** Il est pratique courante de borner la taille totale d'une boucle interne à une constante prédéfinie $K = 30$, ce qui ramène la complexité asymptotique à $\mathcal{O}(n^3)$.
- **Réponse 2 :** Il est possible de traiter séparément la partie symétrique et l'*excroissance* asymétrique, comme illustré dans la figure ci-dessus. Ceci ne permet cependant pas de capturer entièrement tous les aspects du modèle.

En pratique, les paramètres expérimentaux ne sont disponibles que jusqu'à une certaine taille, puis sont extrapolés. Le modèle d'énergie peut donc être approché jusqu'à n'importe quelle précision par un algorithme en $\mathcal{O}(n^3)$.

Cette discussion illustre l'intrication entre modèle d'énergie et modélisation de l'espace de recherche.

Décomposition Mfold/Unafold

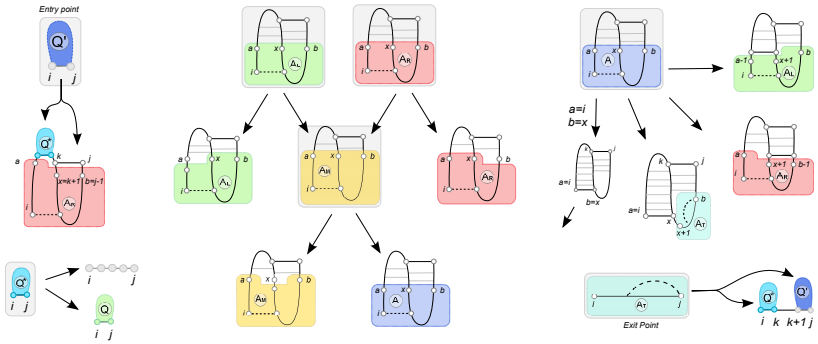


Cette décomposition est non-ambiguë et complète, et donne les arcs d'un F-graphe a $\Theta(n^2)$ sommets et $\mathcal{O}(n^3)$ qui est :

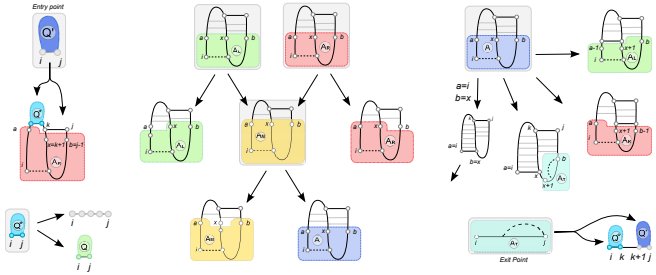
- Acyclique : Largeur d'intervalle strictement décroissante le long d'un arc.
- Independant : Intervalles disjoints deux-à-deux en sortie d'un arc.

Application	Algorithme	Temps	Mémoire	Référence
Minimisation d'énergie	Score Min + $\pi_{\mathcal{T}}$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	[ZS81]
Fonction de partition	Comptage + $e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	[McC90]
Probas paires de bases	Proba arcs + $e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	[McC90]
Échantillonnage statistique (k-samples)	Gen. aléa. + $e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$\mathcal{O}(n^3 + k \cdot n \log n)$	$\mathcal{O}(n^2)$	[DL03, Pon08]
Moments de l'énergie (Moyenne, Var.)	Moments + $e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	[MMN05]
k-ième moment de valuations additives	Moments + $e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$\mathcal{O}(k^3.n^3)$	$\mathcal{O}(k.n^2)$	–
Corrélations de valuations additives	Moments + $e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	–
Moments généraux	Moments + $e^{\frac{-\pi_{\mathcal{T}}}{RT}}$	$\mathcal{O}(4^k.n^3)$	$\mathcal{O}(2^k.n^2)$	–

Basic pseudoknots (Akutsu)



Pseudonoeuds simples (Akutsu)



Décomposition **non-ambiguë** des **pseudonoeuds** simples d'Akutsu *et al.*
$\Rightarrow$ Algorithmes $\mathcal{O}(n^4)/\mathcal{O}(n^4)$ en temps/mémoire pour la minimisation d'énergie en paires de bases, et $\mathcal{O}(n^5)/\mathcal{O}(n^4)$ temps/mémoire pour le modèle de Turner.

On peut directement répondre à des questions telles que :

- Quelle est la probabilité (ie stabilité) de la structure d'énergie minimale parmi les pseudonoeuds simples ?
- Quelle nombre moyen (énergie moyenne et corrélation) de pseudonoeuds dans l'ensemble des repliements d'une séquence ?
- ...

References I

Tatsuya Akutsu.
Dynamic programming algorithms for rna secondary structure prediction with pseudoknots.
Discrete Appl. Math., 104(1-3) :45–62, 2000.

G. Blin, A. Denise, S. Dulucq, C. Herrbach, and H. Touzet.
Alignment of rna structures.
Transactions on Computational Biology and Bioinformatics, ... :..., 2008.
A paraître.

Guillaume Blin, Guillaume Fertin, Irena Rusu, and Christine Sinoquet.
Extending the Hardness of RNA Secondary Structure Comparison.
In Bo Chen, Mike Paterson, and Guochuan Zhang, editors, *ESCAPE'07*, volume 4614 of *LNCS*, pages 140–151, Hangzhou, China, Apr 2007.

S. Cao and S-J Chen.
Predicting structured and stabilities for h-type pseudoknots with interhelix loop.
RNA, 15 :696–706, 2009.

Y. Ding and E. Lawrence.
A statistical sampling algorithm for RNA secondary structure prediction.
Nucleic Acids Res, 31(24) :7280–7301, 2003.

Robert M Dirks and Niles A Pierce.
A partition function algorithm for nucleic acid secondary structure including pseudoknots.
J Comput Chem, 24(13) :1664–1677, Oct 2003.

F. Ferrè, Y. Ponty, W. A. Lorenz, and Peter Clote.
Dial : A web server for the pairwise alignment of two RNA 3-dimensional structures using nucleotide, dihedral angle and base pairing similarities.
Nucleic Acids Research, 35(Web server issue) :W659–668, July 2007.

References II

Claire Herrbach, Alain Denise, and Serge Dulucq.
Average complexity of the jiang-wang-zhang pairwise tree alignment algorithm and of a rna secondary structure alignment algorithm.
In *Proceedings of MACIS 2007, Second International Conference on Mathematical Aspects of Computer and Information Sciences*, 2007.

M. Hochsmann, B. Voss, and R. Giegerich.
Pure multiple RNA secondary structure alignments : A progressive profile approach.
01(1) :53–62, 2004.

Tao Jiang, Lusheng Wang, and Kaizhong Zhang.
Alignment of trees - an alternative to tree edit.
In *CPM '94 : Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, pages 75–86, London, UK, 1994. Springer-Verlag.

R. B. Lyngsøand C. N. S. Pedersen.
RNA pseudoknot prediction in energy-based models.
Journal of Computational Biology, 7(3-4) :409–427, 2000.

J.S. McCaskill.
The equilibrium partition function and base pair binding probabilities for RNA secondary structure.
Biopolymers, 29 :1105–1119, 1990.

D. McLachlan.
Rapid comparison of protein structures.
Acta cristallographica A, 38(6) :871–873, 1982.

István Miklós, Irmtraud M Meyer, and Borbála Nagy.
Moments of the boltzmann distribution for rna secondary structures.
Bull Math Biol, 67(5) :1031–1047, Sep 2005.

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.

References III

Y. Ponty.
Efficient sampling of RNA secondary structures from the boltzmann ensemble of low-energy : The boustrophedon method.
J Math Biol, 56(1-2) :107–127, Jan 2008.

E. Rivas and S.R. Eddy.
A dynamic programming algorithm for RNA structure prediction including pseudoknots.
J Mol Biol, 285 :2053–2068, 1999.

M. Sarver, C. Zirbel, J. Stombaugh, A. Mokdad, and N. B. Leontis.
FR3D : Finding local and composite recurrent structural motifs in RNA 3D.
Journal of Mathematical Biology, 56(1–2) :215–252, January 2008.

S. Wuchty, W. Fontana, I.L. Hofacker, and P. Schuster.
Complete suboptimal folding of RNA and the stability of secondary structures.
Biopolymers, 49 :145–164, 1999.

M. Zuker and P. Stiegler.
Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information.
Nucleic Acids Res, 9 :133–148, 1981.

Application : Implémentation des applications d'hypergraphe en Python