Cours M2 BIBS - Séance 3

Comparaison
Programmation dynamique avancée

Yann Ponty

Bioinformatics Team
École Polytechnique/CNRS/INRIA AMIB – France

23 Janvier 2012

Résumé

1. Alignement et comparaison de structures d'ARN
   - Méthode géométrique
   - Alignement de structures secondaires
   - Méthodes hybrides
2. Programmation dynamique : Méthodes génériques
   - Introduction
   - Analyse syntaxique
   - Hypergraphes

Yann Ponty Cours M2 BIBS - Séance 3 - Comparaison + Prog. dyn. avancée

Résumé

1. Alignement et comparaison de structures d'ARN
   - Méthode géométrique
   - Alignement de structures secondaires
   - Méthodes hybrides
2. Programmation dynamique : Méthodes génériques
   - Introduction
   - Analyse syntaxique
   - Hypergraphes

Yann Ponty Cours M2 BIBS - Séance 3 - Comparaison + Prog. dyn. avancée

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.

Yann Ponty Cours M2 BIBS - Séance 3 - Comparaison + Prog. dyn. avancée

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.

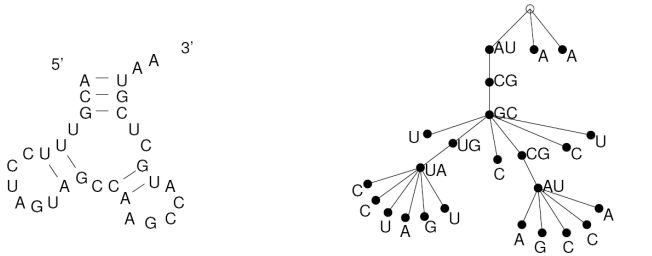
$$L = \sqrt{\min_{R,T} \sum_{i=1}^{m} \|b_i - R(T(m_i))\|^2} \quad A = \sqrt{\sum_{i=1}^{m} \alpha_i^2} \quad D = \frac{1}{m}\sqrt{L^2 + A^2}$$

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) ⇒ Explosion !
Mais recherche exacte réalisable pour des petits motifs.

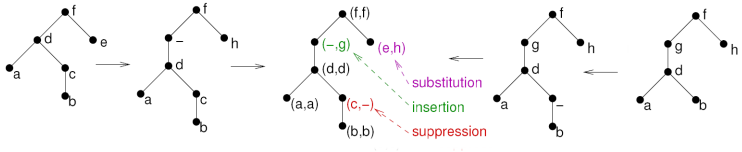
Structures vers arbres

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



Paires de bases ⇒ noeuds internes Bases non-appariées ⇒ 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]

$$\delta(\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle) = \min \begin{cases} \delta(\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle) + \text{del}(\bullet) \\ \delta(\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle) + \text{ins}(\bullet) \\ \delta(\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle) + \text{subst}(\bullet, \bullet) \end{cases}$$

Alignement de forêt

$$\delta(\blacktriangle\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle\blacktriangle) = \min \begin{cases} \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) \mid \blacktriangle\blacktriangle\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\blacktriangle\blacktriangle\} \\ \quad + \text{del}(\bullet) \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle) \mid \blacktriangle\blacktriangle\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\blacktriangle\blacktriangle\} \\ \quad + \text{ins}(\bullet) \\ \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) \end{cases}$$

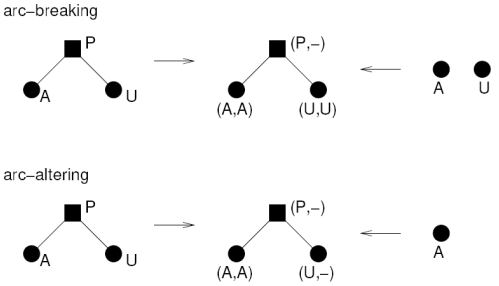
Complexité au pire en $\mathcal{O}(n^4)$ [JWZ94], en moyenne en $\mathcal{O}(n^2)$ [HDD07].
Mais opérations spécifiques à l'ARN manquantes.

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]

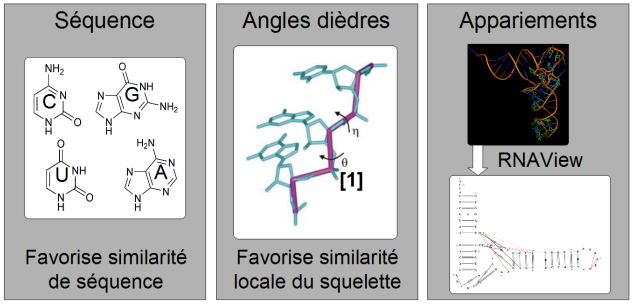
$\delta(\blacktriangle\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle\blacktriangle) =$

$$\min \begin{cases} \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle\blacktriangle\blacktriangle) + \text{BDel}(\bullet) & \text{si } \bullet \text{ base} \\ \delta(\blacktriangle\blacktriangle\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \text{BIns}(\bullet) & \text{si } \bullet \text{ base} \\ \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \text{BSub}(\bullet, \bullet) & \text{si } \bullet \text{ et } \bullet \text{ bases} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\blacktriangle\blacktriangle\} + \text{PDel}(\bullet) & \text{si } \bullet \text{ paire} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\blacktriangle\blacktriangle\} + \text{PIns}(\bullet) & \text{si } \bullet \text{ paire} \\ \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \text{PSub}(\bullet, \bullet) & \text{si } \bullet \text{ et } \bullet \text{ paires} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle^{\bullet}\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\} + \text{Fus}(\bullet, \bullet, \bullet) & \text{si } \bullet \text{ paire et } \bullet \text{ base} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle^{\bullet}\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\} + \text{Sci}(\bullet, \bullet, \bullet) & \text{si } \bullet \text{ paire et } \bullet \text{ base} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\} + \text{GAlt}(\bullet, \bullet) & \text{si } \bullet \text{ paire et } \bullet \text{ base} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle^{\bullet}\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\blacktriangle\blacktriangle\} + \text{DAlt}(\bullet, \bullet) & \text{si } \bullet \text{ paire} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\} + \text{GComp}(\bullet, \bullet) & \text{si } \bullet \text{ paire et } \bullet \text{ base} \\ \min\{\delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) + \delta(\blacktriangle\blacktriangle, \blacktriangle\blacktriangle) : \blacktriangle\blacktriangle^{\bullet}\blacktriangle\blacktriangle = \blacktriangle\blacktriangle\blacktriangle\blacktriangle\} + \text{DComp}(\bullet, \bullet) & \text{si } \bullet \text{ paire} \end{cases}$$

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.

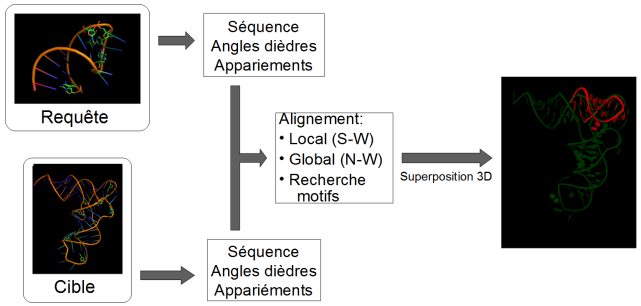
DIAL capture les similarités locales à trois niveau :



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, ...

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).

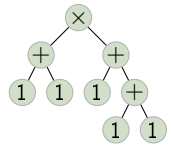
- Approche Hypergraphe :
  Espace de recherche = (hyper-)chemins d'un hypergraphe.

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 :
  $$1 + 1 \times 1 + 1 + 1 = \ ?$$

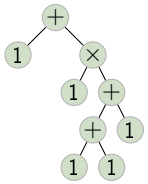
Deux grandes familles de formalismes



- 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 :
  $$1 + 1 \times 1 + 1 + 1 = \ = (1 + 1) \times (1 + (1 + 1)) = 6\,?$$

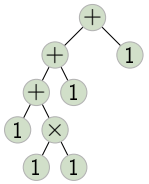
Deux grandes familles de formalismes



- 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 :
$1 + 1 \times 1 + 1 + 1 = \; = 1 + (1 \times ((1 + 1) + 1)) = 4$?

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).

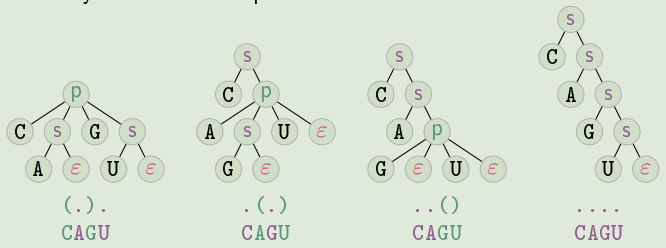
Deux grandes familles de formalismes

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

Ex2 (ARN) :

$R \to \mathtt{A}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{A}\,R \mid \mathtt{G}\,R\,\mathtt{C}\,R \mid \mathtt{C}\,R\,\mathtt{G}\,R \mid \mathtt{G}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{G}\,R$ (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 :



Deux grandes familles de formalismes

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

Ex2 (ARN) :

$R \to \mathtt{A}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{A}\,R \mid \mathtt{G}\,R\,\mathtt{C}\,R \mid \mathtt{C}\,R\,\mathtt{G}\,R \mid \mathtt{G}\,R\,\mathtt{U}\,R \mid \mathtt{U}\,R\,\mathtt{G}\,R$ (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)

Deux grandes familles de formalismes

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

  Ex2 (ARN) :

  $R \rightarrow AURU \mid URAR \mid GRCR \mid CRGR \mid GRUR \mid URGR$ (Règles p)
  $\mid AR \mid CR \mid GR \mid UR \mid \varepsilon$ (Règles s)

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

  Intérêt :
  - Algorithmique générique en $O(n^3)$
  - Pas de manipulation des indices ⇒ 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

Deux implémentations de l'analyses syntaxique générique

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

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

[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).

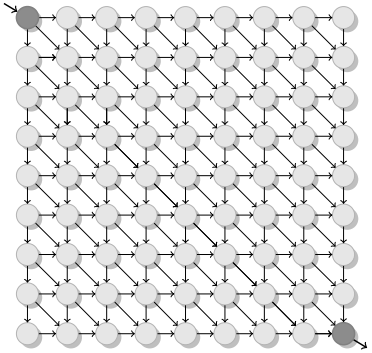
Pourquoi les hypergraphes ?

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	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

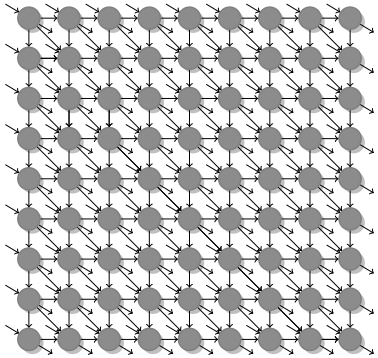
Pourquoi les hypergraphes ?

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.



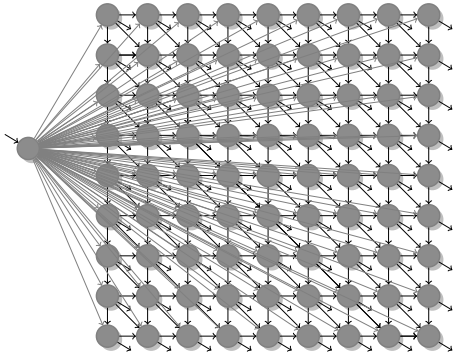
Pourquoi les hypergraphes ?

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.



Pourquoi les hypergraphes ?

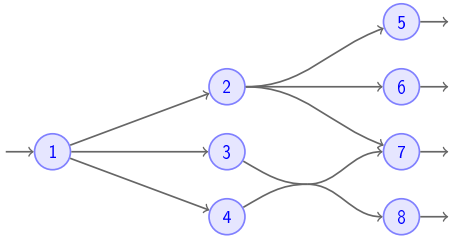
Rappel : Dans Smith-Waterman, la remontée (ou backtrack) correspond à un chemin de coût minimal dans la matrice, ie un chemin dont la somme des coûts des arêtes est minimale.



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



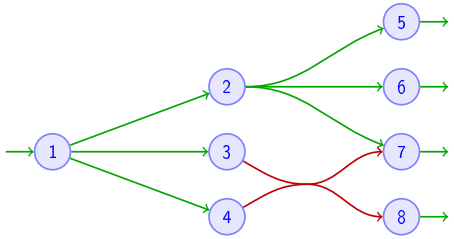
Les hypergaphes généralisent les graphes (dirigés) à des (hyper) arcs de degrés entrant/sortant arbitraires.

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 $e = (t(e) \to h(e))$ tels que $t(e), h(e) \subset E$

Hypergraphes



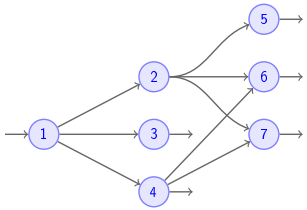
Les hypergaphes généralisent les graphes (dirigés) à des (hyper) arcs de degrés entrant/sortant arbitraires.

Definition (Hypergraphes)

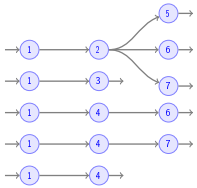
Un hypergraphe $\mathcal{H}$ est un couple (V, E) tel que :

- V est un ensemble de sommet
- E est un ensemble d'hyperarcs $e = (t(e) \to h(e))$ 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

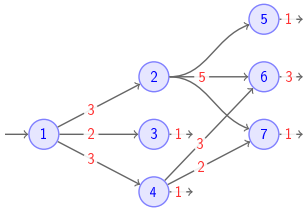


Tous les F-chemins partant du sommet 1

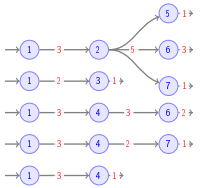
Definition (F-chemin)

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.



F-graphe



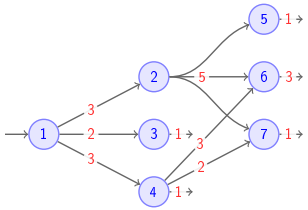
Tous les F-chemins partant du sommet 1

Definition (F-chemin)

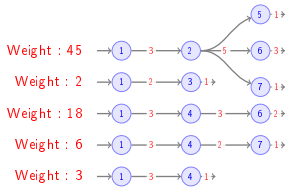
Un F-chemin est un arbre de racine $s \in V$, et dont les fils sont des F-chemins construits à partir des sommets sortants d'un arc $e = (s \to \mathbf{t}) \in E$.

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.



F-graphe



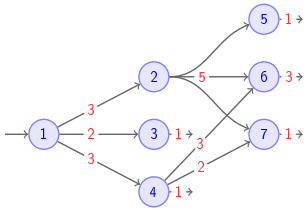
Tous les F-chemins partant du sommet 1

Definition (F-chemin)

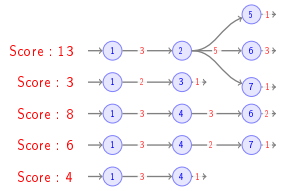
Un F-chemin est un arbre de racine $s \in V$, et dont les fils sont des F-chemins construits à partir des sommets sortants d'un arc $e = (s \to \mathbf{t}) \in E$.

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.

Algorithmes basiques

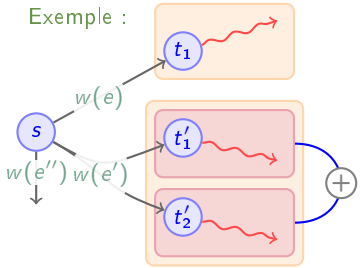
$\mathcal{H} = (s_0, V, E, \pi)$: hypergraphe acyclique s_0 : Sommet initial π : valuation

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.

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

Exemple :



Algorithmes basiques

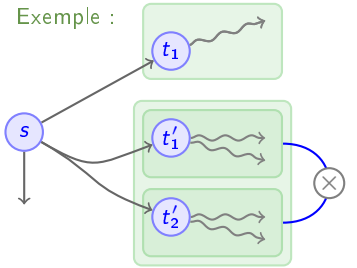
$\mathcal{H} = (s_0, V, E, \pi)$: hypergraphe acyclique s_0 : Sommet initial π : valuation

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.

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

Exemple :



Algorithmes basiques

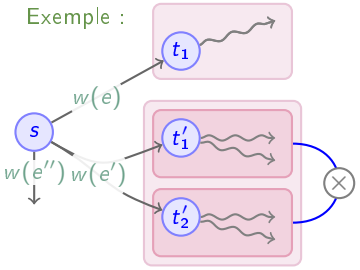
$\mathcal{H} = (s_0, V, E, \pi)$: hypergraphe acyclique s_0 : Sommet initial π : valuation

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 :

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

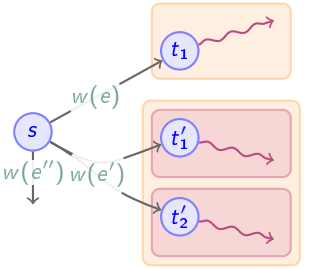
De nouvelles questions *ensemblistes* se posent alors :

- 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.
- 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 :

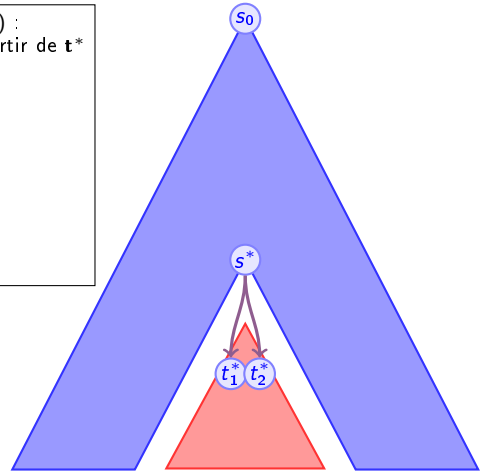
$$\mathbb{P}(p|\pi) = \frac{\prod_{e \in p} \pi(e)}{w_{s_0}}, \forall p \in \mathcal{T}.$$

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

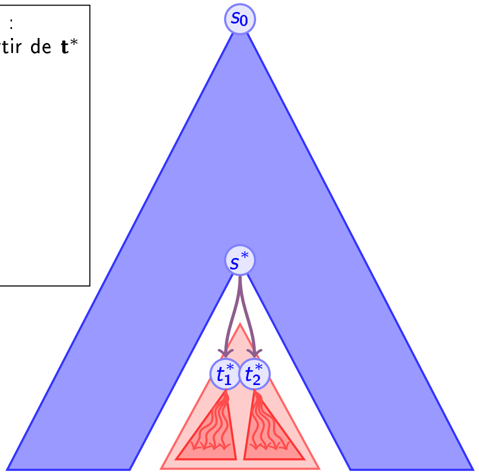
F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$



Algorithme Inside/outside

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

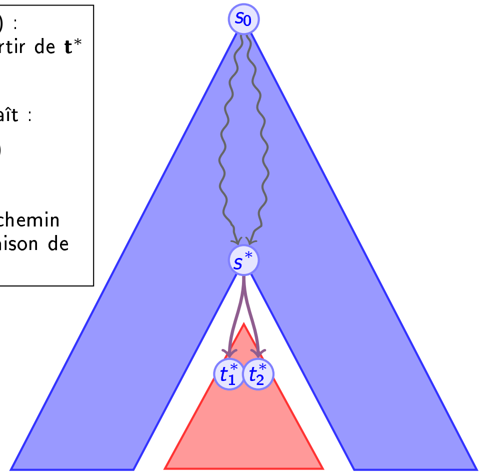
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$



Algorithme Inside/outside

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

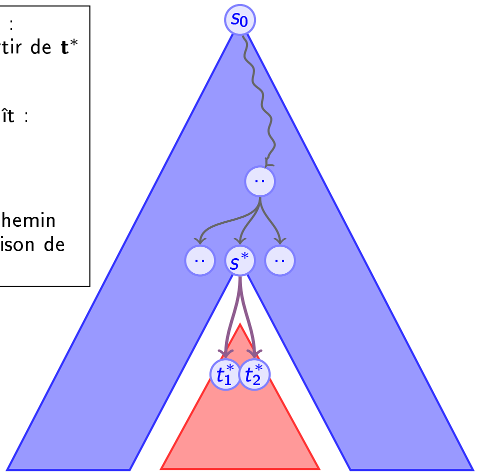
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : Contexte où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, compléter le F-chemin en explorant toute combinaison de frêres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



Algorithme Inside/outside

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

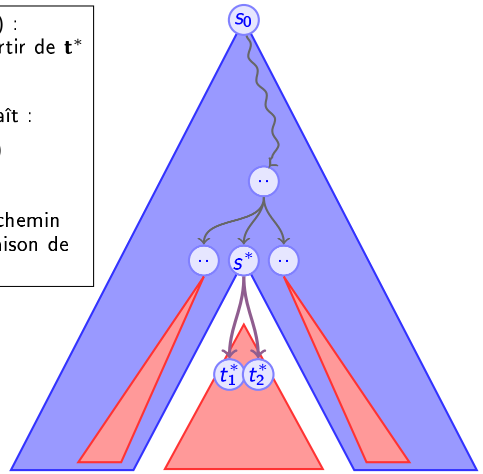
- Inside : $|t^*|$-uplet construit à partir de $\mathbf{t}^*$
- Arc distingué e^*
- Outside : Contexte où e^* apparaît :
  - Chemin $p = (s_0, s_1, \ldots, s^*)$ using F-arcs $(e_1, \ldots, e_k)$
  - Pour chaque sommet dans $h(e_i)/\{p\}$, compléter le F-chemin en explorant toute combinaison de frêres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



Algorithme Inside/outside

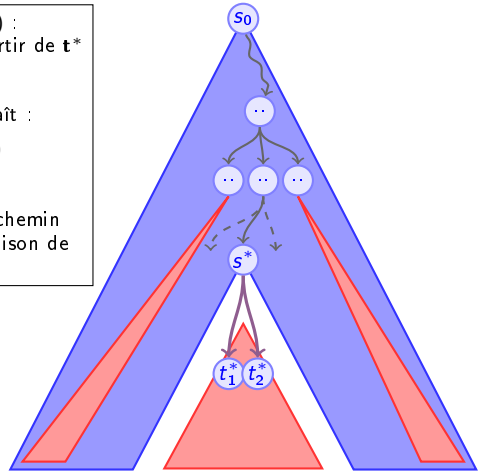
F-chemin empruntant $e^* = (s^* \to \mathbf{t}^*)$:

- 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 frêres pour p.



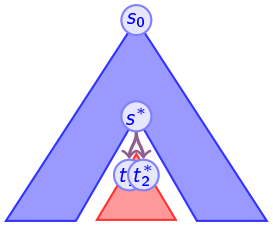
Algorithme Inside/outside

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 :

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

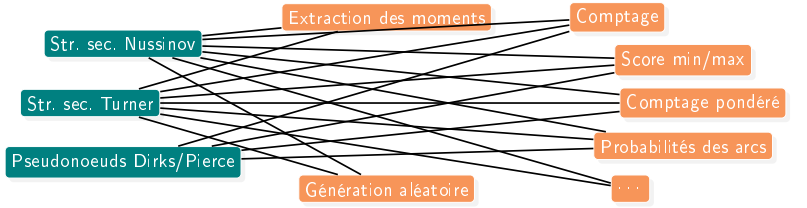
De nouvelles questions ensemblistes se posent alors :

- 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

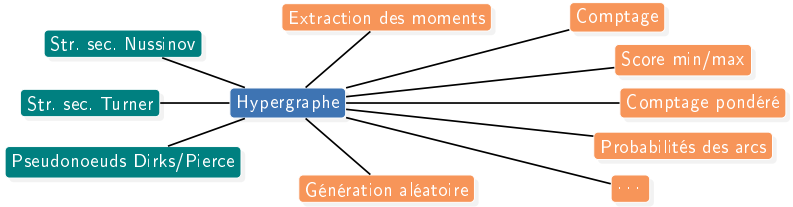
Applications spécifiques de la programmation dynamique peuvent (et doivent) être détachées de l'équation, et être exprimée à un niveau abstrait.



Mi-temps

Message #1

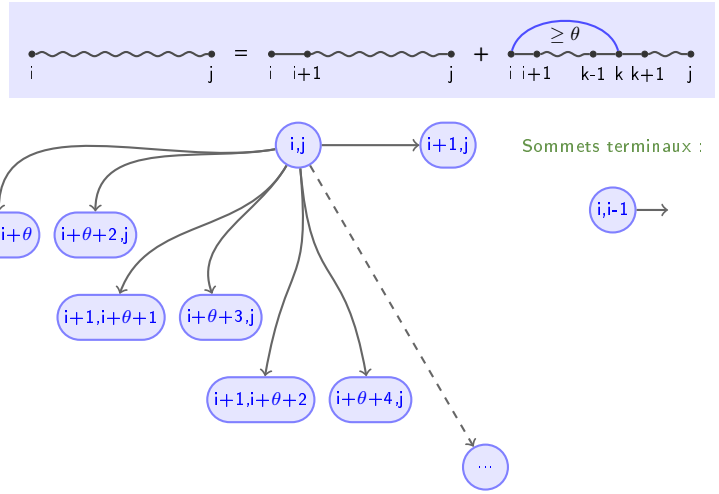
Applications spécifiques de la programmation dynamique peuvent (et doivent) être détachées de l'équation, et être exprimée à un niveau abstrait.



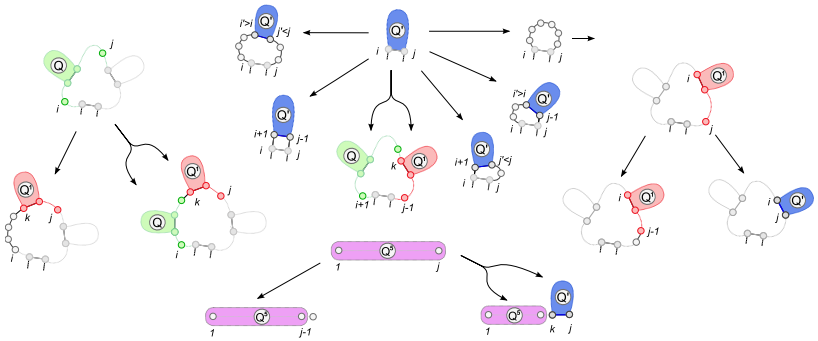
Credits : Roytberg and Finkelstein pour le *parallèle* Hypergraphe en Bioinformatique, L. Hwang pour une jolie formalisation algébriques de la programmation dynamique hypergraphe, Flajolet *et al* pour des transformations symboliques utilisé pour l'extraction des moments ...

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

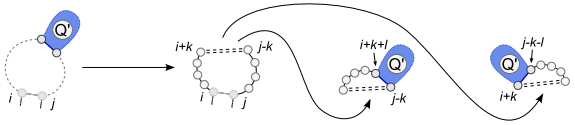


Figure: Stratégies alternatives pour les boucles internes, créant tout d'abord la partie symétrique en premier, puis la partie asymétrique.

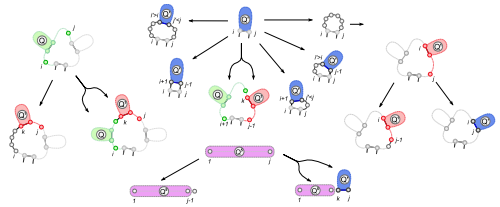
Du fait des boucles internes, l'ensemble des F-arcs engendrés pour le cas Q' est de cardinalité en $\mathcal{O}(n^4)$, ce qui contredit la complexité annoncée en $\mathcal{O}(n^3)$. Deux niveaux de réponses à cela :

- 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

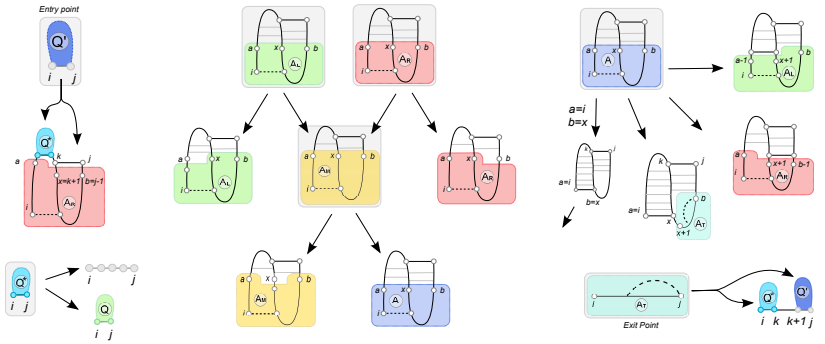


Cette décomposition est non-ambiguë et complète, et donne les arcs d'un F-graphe a $\Theta(n^2)$ sommets et $\mathcal{O}(n^3)$ qui est :

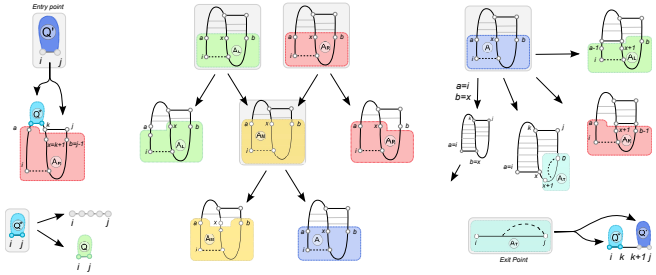
- Acyclique : Largeur d'intervalle strictement décroissante le long d'un arc.
- Independant : Intervalles disjoints deux-à-deux en sortie d'un arc.

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.*
⇒ 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

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.

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

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.

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.

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.

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.

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