Alignements de deux séquences

Informatique Génomique - Master 1

Stéphane Vialette

LIGM 8049,
Bureau 4B066
Université Paris-Est Marne La Vallée
vialette@univ-mlv.fr
http://igm.univ-mlv.fr/∼vialette

11 avril 2012

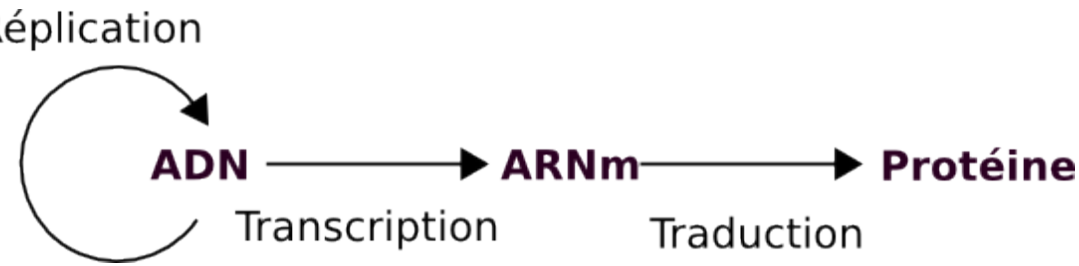
Plan

Plan

Qu'est-ce qu'une séquence ?

Rappels sur les séquences génomiques

- Une séquence génomique est l'enchaînement des nucléotides le long d'une macromolécule d'ADN.
- Représentée par une chaîne de caractères utilisant l'alphabet {A,C,G,T} - qui distingue les quatre types de nucléotides.
- C'est cette séquence qui, après transcription en ARNm puis traduction, conduira à une protéine.
- Une séquence protéique est l'enchaînement des vingt types d'acides aminés le long d'un polypeptide.
- Représentée par une chaîne de caractères sur un alphabet de vingt lettres.



Qu'est-ce qu'une séquence ?

Rappels sur les séquences génomiques

- La longueur de la séquence complète d'un génome bactérien est de l'ordre du million de paires de bases ;
- Celle d'un génome eucaryote est typiquement plus longue de deux ou trois ordres de grandeur (en milliards).
- Un gène bactérien a pour ordre de grandeur le millier ;
- Celle d'un gène eucaryote est supérieure d'un ordre de grandeur en moyenne, mais peut atteindre le million.
- Une séquence protéique comporte une centaine de caractères.

Pourquoi comparer des séquences ?

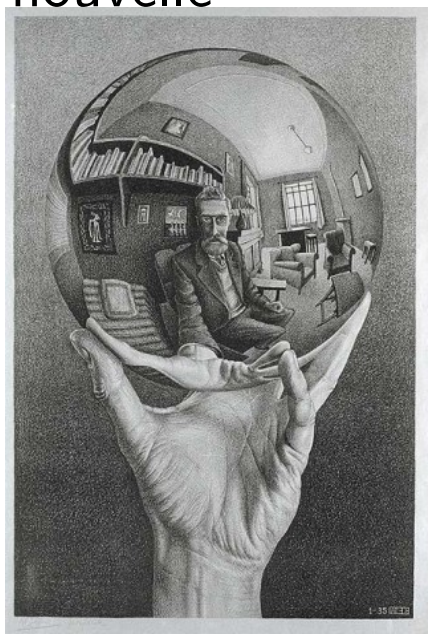
La comparaison de séquences

- La tâche informatique la plus fréquemment exécutée par les biologistes.
- Déterminer les similitudes éventuelles entre deux séquences génomiques ou protéiques.
- Le but : inférer des connaissances sur une nouvelle séquence à partir des connaissances sur d'autres séquences proches
  - ce qui se ressemble s'assemble ...
  - si la fonction d'une séquence est connue, la fonction de la seconde peut s'en déduire.

Pourquoi comparer des séquences ?

La comparaison de séquences

- Les connaissances sur ces séquences sont stockées dans des bases de données.
- Le premier réflexe d'un biologiste est de parcourir ces bases de données à la recherche de séquences similaires à la nouvelle séquence qu'il détient.
- La comparaison de séquences permet également
  - de prédire des gènes
  - de déterminer la fonction d'une protéine
  - de prédire des structures
  - ...



D'où viennent les similitudes entre séquences ?

Origine de la similitude de séquences

- Les modifications de la séquence génomique découlent de :
  - la substitution d'un nucléotide par un autre
  - la disparition d'un nucléotide
  - l'apparition d'un nucléotide
- Ces erreurs/mutations se propagent au sein des populations par héritage génétique.
- La séquence d'un génome d'une espèce, présente au sein de ses chromosomes, évolue dans le temps.

D'où viennent les similitudes entre séquences ?

Histoire des espèces

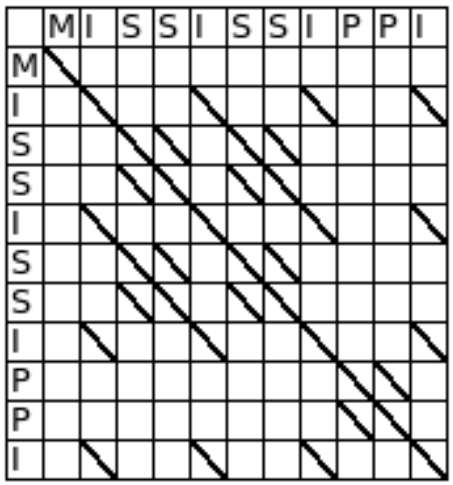
- Représentable par un arbre dont les feuilles sont les espèces actuelles.
- Deux espèces sont considérées d'autant plus proches que leur espèce ancestrale commune est récente.

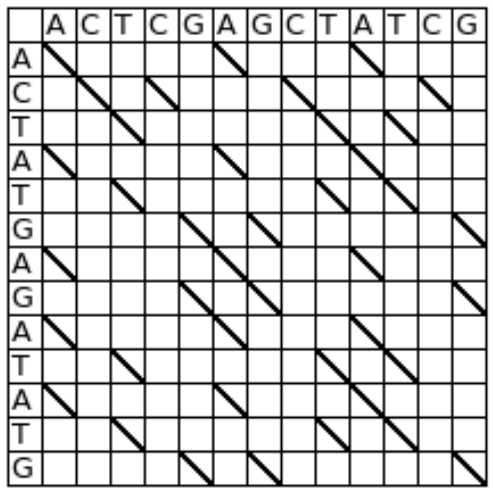
Plan

Les mesures de similarité

Vers une première solution - Dotplot

- En 1982, Staden a proposé une méthode visuelle pour comparer deux séquences : la matrice de points (dotplot).
- Une séquence – disons S – est positionnée en ligne, l'autre – disons T – en colonne dans une matrice.
- Toute case (i, j) de la matrice est noircie si $S[i] = T[j]$.
- Dans cette représentation, une diagonale correspond à une zone de similarité.

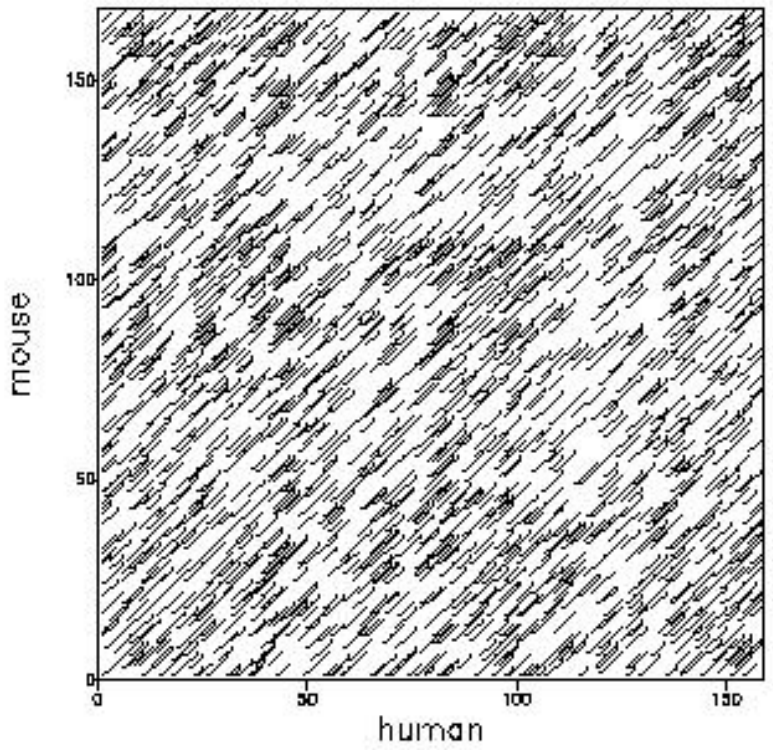




Les mesures de similarité

Limites de Dotplot

- Principalement, la lisibilité !
- Les petites similarités créent du *bruit*.
- Il existe des techniques pour le limiter
- Il suffit par exemple de ne considérer que les diagonales de taille $> k$



Les mesures de similarité

Distance de Hamming

- C'est la mesure de similarité triviale entre deux séquences de même longueur.
- Formellement : $d(a, b) = \sum_{i=0}^{n-1}(a[i] \oplus b[i])$

Exemple

Comparaison simple d'une paire de séquences

- *(a)* `AGTATC` *et* `AGATGC`*; 3 différences*
- *(b)* `AGTTTC` *et* `AGATTC`*; 1 différence*

Les séquences de la paire (b) sont considérées comme étant plus similaires que celles de la paire (a) si on considère la distance de Hamming.

Les mesures de similarité

Limite de la distance de Hamming

- Les séquences à comparer ont rarement la même longueur
- Même si c'est le cas, rien ne dit qu'elles doivent être comparées sur cette longueur exactement.
- Dans le cadre de séquences génomiques, des nucléotides ou des acides aminés ont pu s'insérer ou au contraire disparaître au cours de l'évolution.

Distance de Levenshtein

- Généralisation de la distance de Hamming.
- Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer, ou remplacer pour passer d'une chaîne à l'autre
- Egalement appelée distance d'édition.

Alignement et distance d'édition

Alignement et gaps

- Alignement = mise en correspondance de 2 séquences lettre à lettre.
- Pour tenir compte des insertions ou délétions éventuelles, il faut introduire un caractère particulier, appelé *gap* et noté $-$.
- L'alignement d'un caractère avec un gap s'interprète soit comme une insertion du caractère dans la première séquence, soit comme une délétion d'un caractère dans la seconde.

Exemple

$$(a)\quad \begin{matrix} AG\text{-}ATGCT \\ AGTAT\text{-}C\text{-} \end{matrix} \qquad (b)\quad \begin{matrix} AGAT\text{-}GCT \\ AG\text{-}TATC\text{-} \end{matrix}$$

Coût d'un alignement

Matrice de substitution

- C'est un tableau contenant les coûts de la substitution (*i.e.* de mise en correspondance) d'un caractère par un autre.
- Cette matrice est de taille $|\Sigma|x|\Sigma|$ pour un alphabet Σ

Exemple

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

Coût d'un alignement

Calcul du coût

- Il faut fixer le coût d'insertion d'un gap (appelé *indel*).
- Le coût d'un alignement (S', T') de S et T correspond alors à $cout(S', T') = \sum_{i=0}^{|S'|-1} \alpha(S'[i], T'[i])$ *tq*

$$\alpha(S'[i], T'[i]) = \begin{cases} sub, & si\ S'[i] \neq T'[i] \\ indel, & S'[i]\ ou\ T'[i]\ est\ un\ gap \\ 0, sinon & \end{cases}$$

Exemple

Avec $indel = 2$ *et* $sub = 1$,

S'	AG-ATGCT	AGAT-GCT
T'	AGTAT-C-	AG-TATC-
Coût	6	7

Alignement optimal

Calcul des alignements optimaux

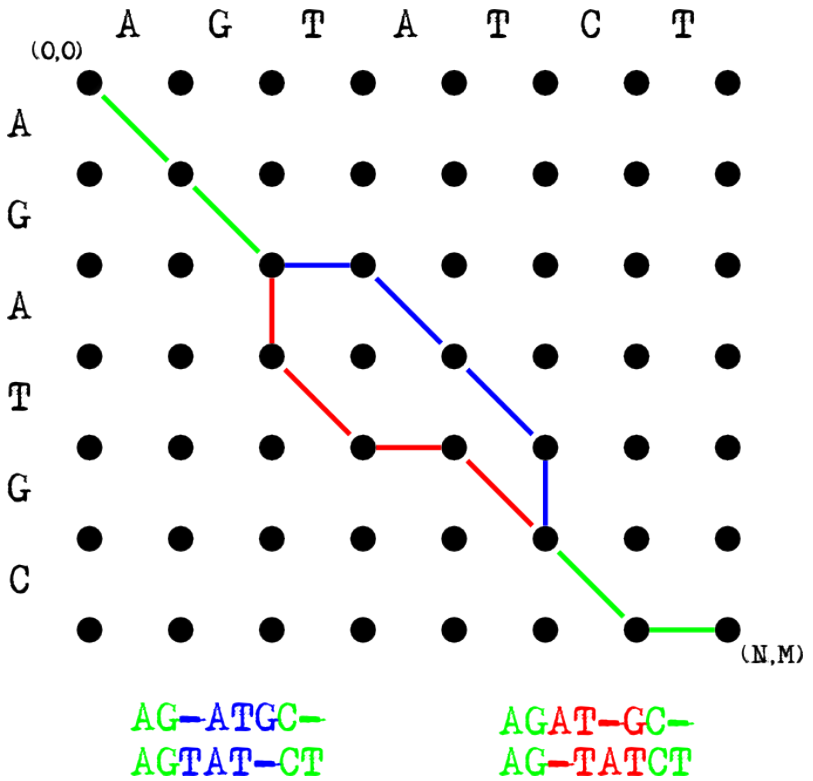
- Parmi tous les alignements possibles, le principe est de retenir ceux (il peut y en avoir plusieurs) dont le coût est minimal.
- Un alignement de coût minimal est dit optimal.
- La similarité des deux séquences se mesure alors par ce coût minimal.
- Il existe une autre mesure proche : le score d'alignement où les indels et les substitutions ont un coût négatif et les appariements (ou match) un coût positif.
- On cherche alors l'alignement de score maximal.

Plan

Représentation graphique d'un alignement

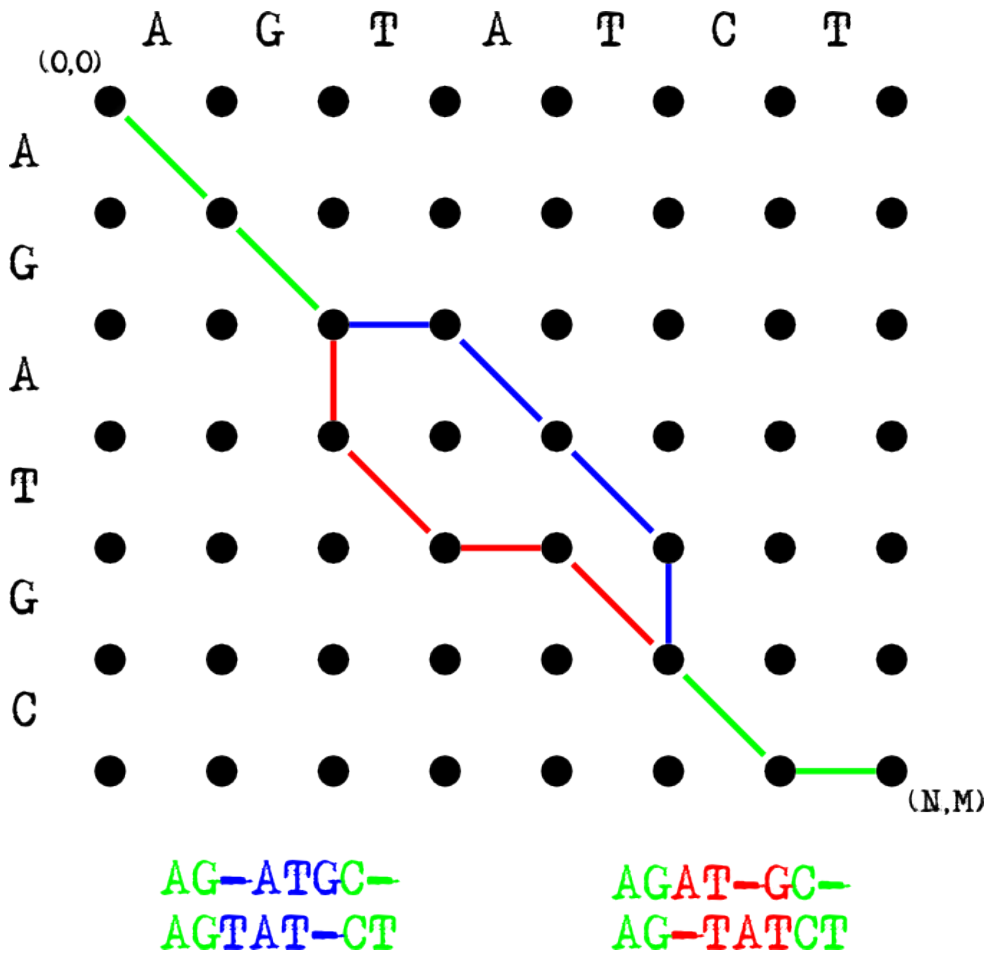
Graphe d'édition

- Tout alignement peut se représenter sous la forme d'un graphe.
- Etant données deux séquences de longueur N et M, tout alignement de ces séquences correspond à un chemin menant du noeud (0,0) au noeud (N,M) d'une grille où chaque colonne correspond à une lettre de la première séquence et chaque ligne à une lettre de la seconde.



Représentation graphique d'un alignement

Graphe d'édition



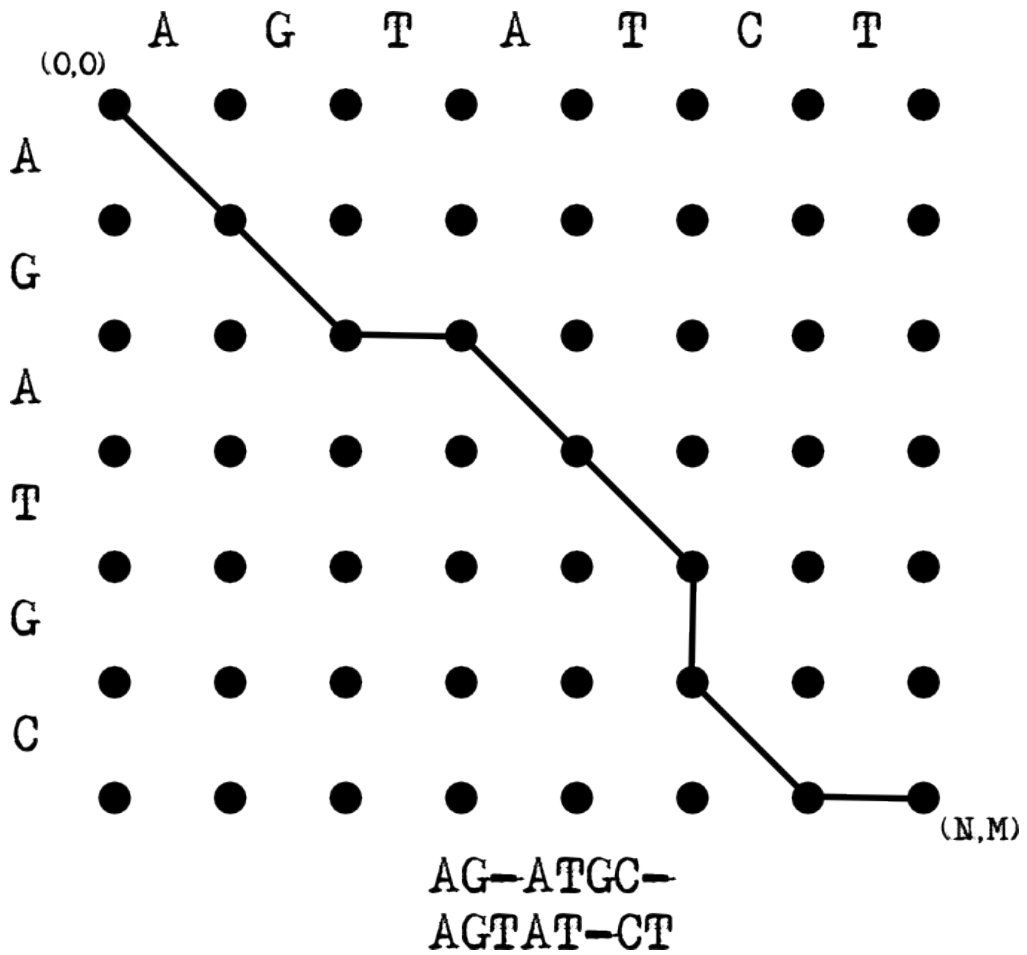
Algorithme de recherche

Propriété d'optimalité

- Soit C_L un chemin optimal du noeud (0,0) au noeud (N,M) et de longueur L.
- Le sous-chemin C_{L-1} de longueur L-1, composé des L-1 premiers segments de C_L correspond à l'alignement complet privé de l'alignement de la dernière paire de caractères.
- C_{L-1} est optimal ; preuve par l'absurde.
  - Hyp. : C_{L-1} n'est pas optimal mais C_L l'est.
  - On en déduit qu'il existe un autre sous-chemin de longueur L-1 s'achevant sur le même noeud que C_{L-1} avec un coût inférieur.
  - Dans ce cas, C_L ne serait pas optimal ; une contradiction.
- Par récurrence, on en déduit que tous les sous-chemins d'un chemin optimal sont optimaux.

Algorithme de recherche

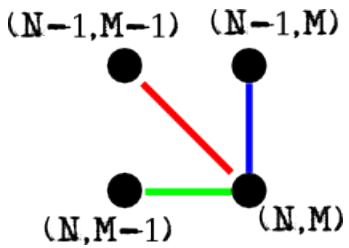
Propriété d'optimalité



Algorithme de recherche

Algorithme

- Un alignement (ou chemin) optimal de longueur L s'obtient donc en complétant un alignement partiel (ou sous-chemin) optimal de longueur L-1.
- Or, il n'existe que trois sous-chemins de longueur L-1 du chemin arrivant au noeud (N,M) :
  - celui venant du noeud (N,M-1) ;
  - celui venant du noeud (N-1,M-1) et
  - celui celui venant du noeud (N-1,M).



Exemple

ACGGCTA T ou *ACGGCTA* T ou *ACGGCTAT* −
????? | ????? ?????
ACTGTA T *ACTGTAT* − *ACTGTA* T

Algorithme de recherche

Formule de récurrence

On note $Cout(i, j)$ le coût optimal entre $S[0..i]$ et $T[0..j]$

- $Cout(0, 0) = 0$
- $Cout(0, j) = Cout(0, j-1) + indel$
- $Cout(i, 0) = Cout(i-1, 0) + indel$
- $$Cout(i, j) = min \begin{cases} Cout(i-1, j-1) + sub(S[i], T[j]) \\ Cout(i, j-1) + indel \\ Cout(i-1, j) + indel \end{cases}$$

Que se passe-t-il si on implémente ces formules de manière récursive ?

Plan

Alignement optimal

Programmation dynamique

- Lors de la démarche récursive de détermination du coût du chemin optimal arrivant sur un noeud, le coût d'un même sous-chemin est calculé plusieurs fois, pénalisant lourdement l'efficacité de l'algorithme.
- La programmation dynamique propose de stocker des résultats intermédiaires dans une table (dite de programmation dynamique).
- Le coût en espace mémoire est largement amorti par le gain en temps.
- Solution proposée par Neddleman et Wunsch.

Alignement optimal

L'alignement global

Evaluation d'une ressemblance globale entre deux séquences.

- Données :
  - Deux séquences
  - Des coûts pour les opérations d'édition
- Problème :
  - Trouver un alignement optimal ?

Alignement optimal

1ere étape

- Consiste à créer une table indéxée par les deux séquences
- Chaque case de la table correspond à un noeud du graphe d'édition
- On stocke dans la case (i, j) le coût d'alignement des i premières bases de S et des j premières bases de T.
- L'initialisation consiste à remplir la 1ère ligne et la 1ère colonne.
- On remplit ligne à ligne.

Visualiser l'algorithme ...

Alignement optimal

1ere étape

- Consiste à créer une table indéxée par les deux séquences

		A	G	T	A	T	C	T
	0	1	2	3	4	5	6	7
A	1							
G	2							
A	3							
T	4							
G	5							
C	6							

- Initialisation consiste à remplir la 1ère ligne et la 1ère colonne.

Alignement optimal

1ere étape

- Consiste à créer une table indéxée par les deux séquences

		A	G	T	A	T	C	T
	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
G	2	1	0	1	2	3	4	5
A	3							
T	4							
G	5							
C	6							

- On remplit ligne à ligne.

Alignement optimal

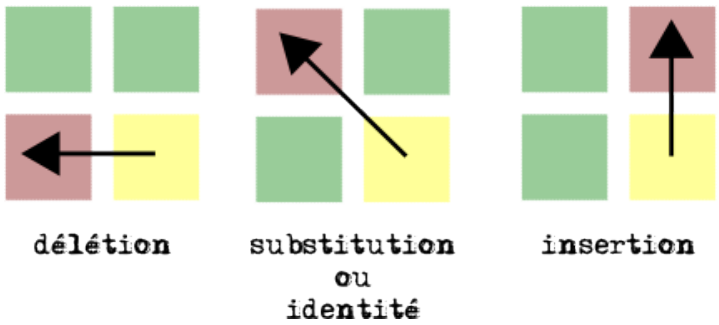
1ere étape

- Le résultat de cette 1ere étape est donc un tableau à $N+1$ lignes et $M+1$ colonnes qui contient en case (i, j) le coût du meilleur alignement impliquant les i premiers caractères de la première séquence et les j premiers de la seconde.
- La valeur au noeud (N, M) correspond au coût de tout alignement optimal des séquences complètes.
- Mais ce tableau ne dit pas explicitement quels sont les alignements optimaux.

Alignement optimal

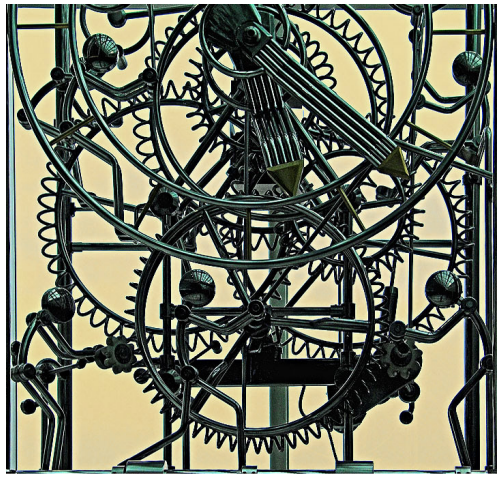
2nde étape

- Pour connaître l'alignement optimal, il faut appliquer un algorithme de *backtracking*.
- Cela correspond à appliquer un cheminement "en arrière" de (N, M) en choisissant à chaque fois la direction qui ramène au noeud jusqu'à arriver en $(0, 0)$
- Le chemin ainsi tracé fournit l'alignement.



Visualiser l'algorithme ...

Alignement optimal



Complexité de l'algorithme

- Pour le calcul du score d'alignement :
  - (Etape 1) – $O(nm)$ en temps, $O(min\{n, m\})$ [1] en espace
- Pour la construction de l'alignement :
  - (Etapes 1 et 2) – $O(nm)$ en temps et en espace

1. Deux lignes ou colonnes sont effectivement nécessaires

Alignement optimal

L'alignement local

Evaluation d'une ressemblance locale entre deux séquences.

- Données :
  - Deux séquences
  - Des coûts pour les opérations d'édition
- Problème :
  - Trouver un alignement optimal ?

La ressemblance étant locale, on recherche l'alignement optimal de sous-séquences

Alignement optimal

Formule de récurrence

On note $Score(i, j)$ le coût optimal entre $S[0..i]$ et $T[0..j]$

- $Score(0, 0) = \mathbf{Score(0, j)} = \mathbf{Score(i, 0)} = 0$
- $$Score(i, j) = max \begin{cases} \mathbf{0} \\ Score(i-1, j-1) + sub(S[i], T[j]) \\ Score(i, j-1) + indel \\ Score(i-1, j) + indel \end{cases}$$

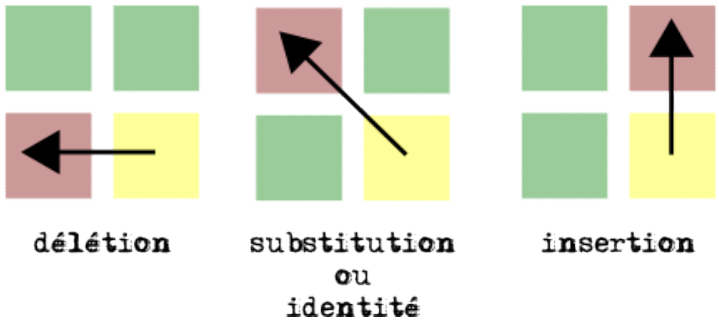
Le 0 permet de débuter l'alignement optimal n'importe où.
L'emploi du 0 implique une notion de similarité (et non plus de distance) :

- $sub(S[i], T[j]) > 0$ si $S[i] = T[j]$, < 0 sinon
- $indel < 0$

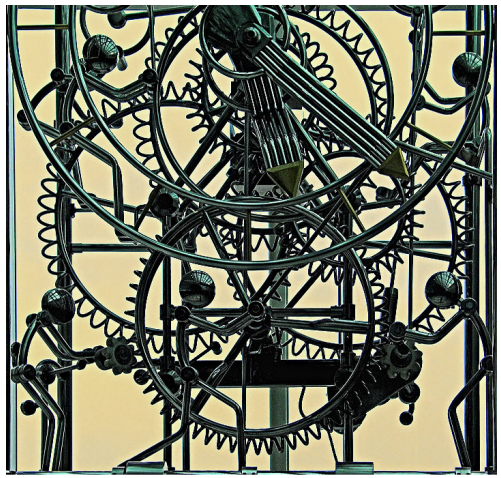
Alignement optimal

L'algorithme

- La première étape consiste également au remplissage de la table de programmation dynamique
- Pour retrouver le score ansi que l'alignement local des deux séquences, il faut trouver la valeur max dans la table puis appliquer un algorithme de backtrace



Alignement optimal



Complexité de l'algorithme

- Pour le calcul du score d'alignement :
  - (Etape 1) – $O(nm)$ en temps, $O(min\{n, m\})$ en espace
- Pour la construction de l'alignement :
  - (Etapes 1 et 2) – $O(nm)$ en temps et en espace

Plan

Optimisation du score

Enrichissement du modèle : fonction de gap affine

- Traitement des gaps consécutifs
  - Une succession de délétions ou d'insertions correspond à un seul évènement mutationnel (insertion ou disparition d'un bloc)

- Le coût d'une succession de k gaps : $indel(k) = g + h(k-1)$
- Coût décomposé en un coût d'*ouverture* (g) et un coût de *propagation* (h)

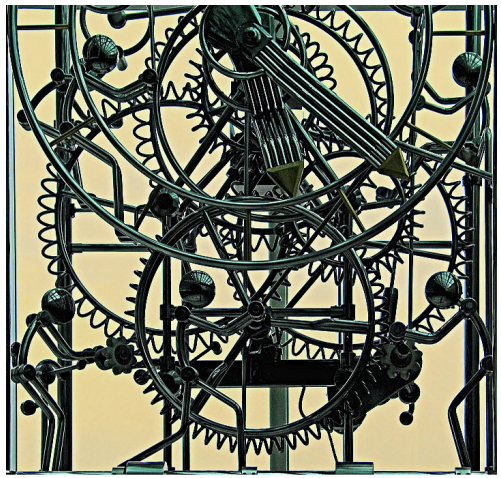
Optimisation du score

Fonction de gap affine

- Il faut trois tables ; $I(nsertion)$, $D(eletion)$, $C(out)$
- On note $I(i,j)$ (resp. $D(i,j)$) $\equiv$ le coût optimal entre $S[0..i]$ et $T[0..j]$ finissant par une insertion (resp. délétion)
- $C(i,j) \equiv$ le coût optimal entre $S[0..i]$ et $T[0..j]$

- $I(i,j) = min \begin{cases} I(i,j-1) + h & (extension) \\ C(i,j-1) + g & (ouverture) \end{cases}$
- $D(i,j) = min \begin{cases} D(i-1,j) + h & (extension) \\ C(i-1,j) + g & (ouverture) \end{cases}$
- $C(i,j) = min \begin{cases} C(i-1,j-1) + sub(S[i], T[j]) \\ D(i,j) \\ I(i,j) \end{cases}$

Optimisation du score



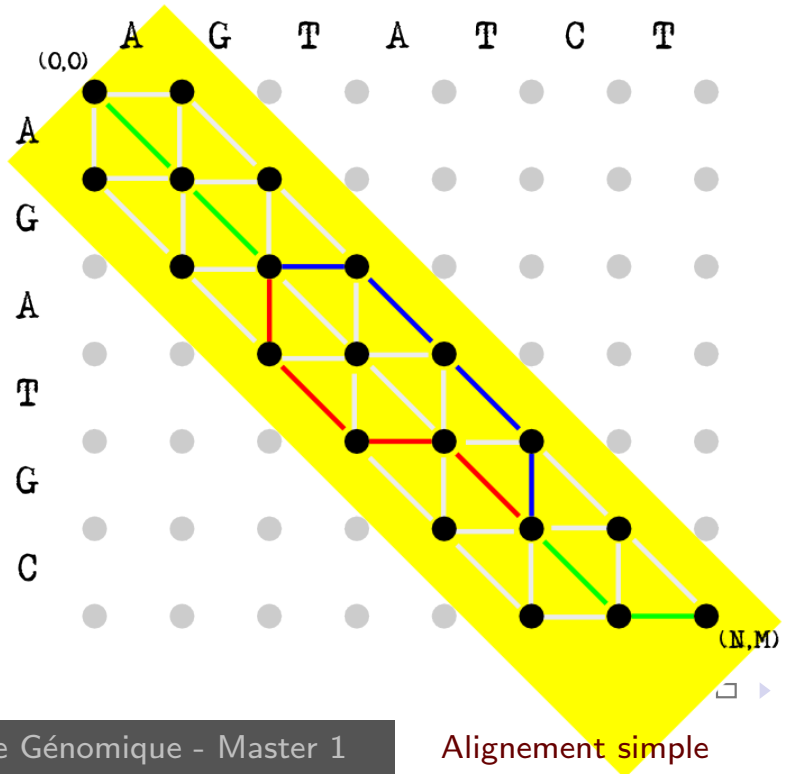
Complexité de l'algorithme

- Pour le calcul du score d'alignement :
  - (Etape 1) – $O(nm)$ en temps, $O(min\{n, m\})$ en espace
- Pour la construction de l'alignement :
  - (Etapes 1 et 2) – $O(nm)$ en temps et en espace

Optimisation du temps

Alignement k-bande

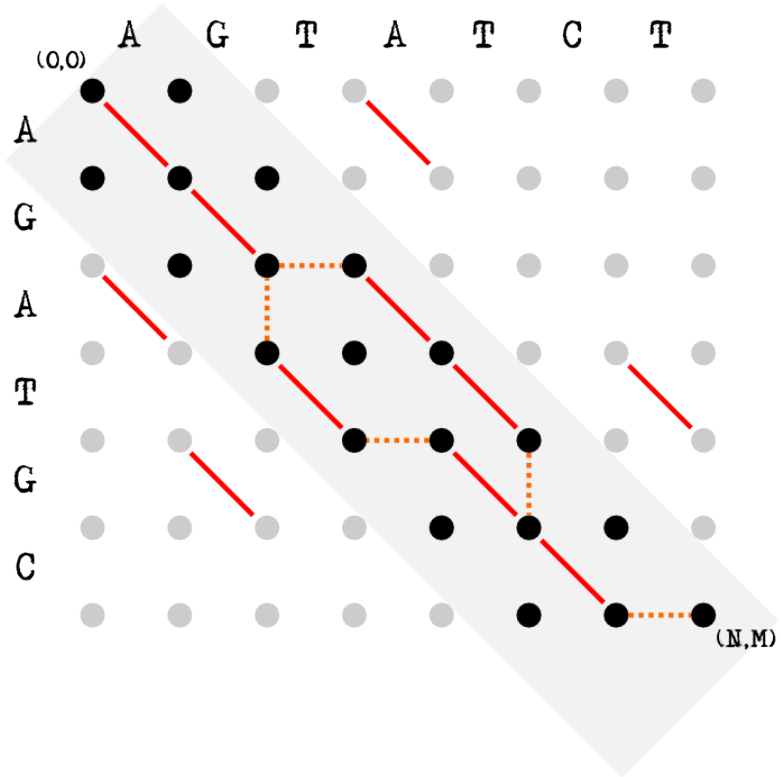
- Par définition, si 2 séquences sont similaires l'alignement optimal sera proche de la diagonale
- On se propose alors de ne calculer qu'une *bande* de largeur k autour de la diagonale



Optimisation du temps

Fasta

- Heuristique basée sur *k*-bande
- L'algorithme :
  1. Recherche des *l*-grams entre S et T (régions locales similaires)
  2. Calcul d'une bande contenant le plus de *l*-grams
  3. Calcul de l'alignment optimal au sein de cette bande



Optimisation du temps

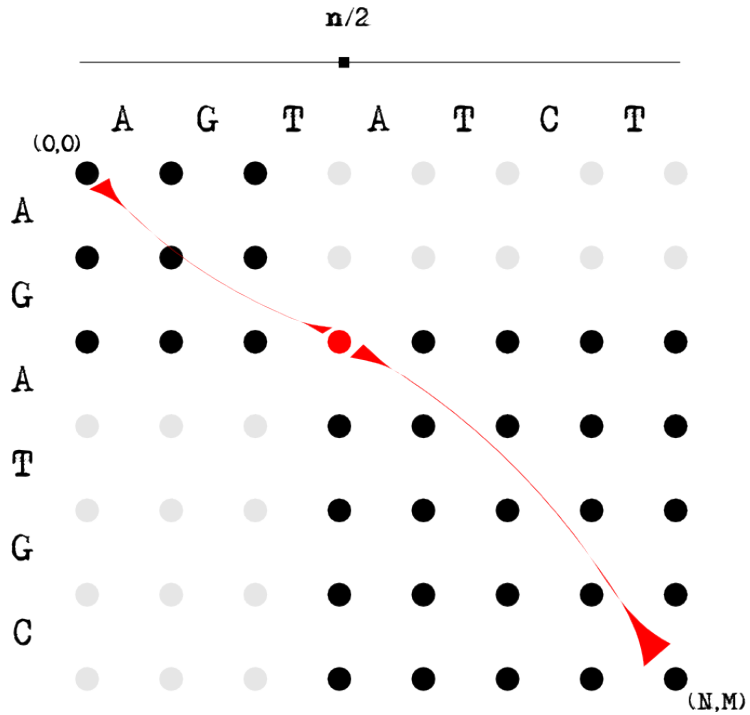
Blast

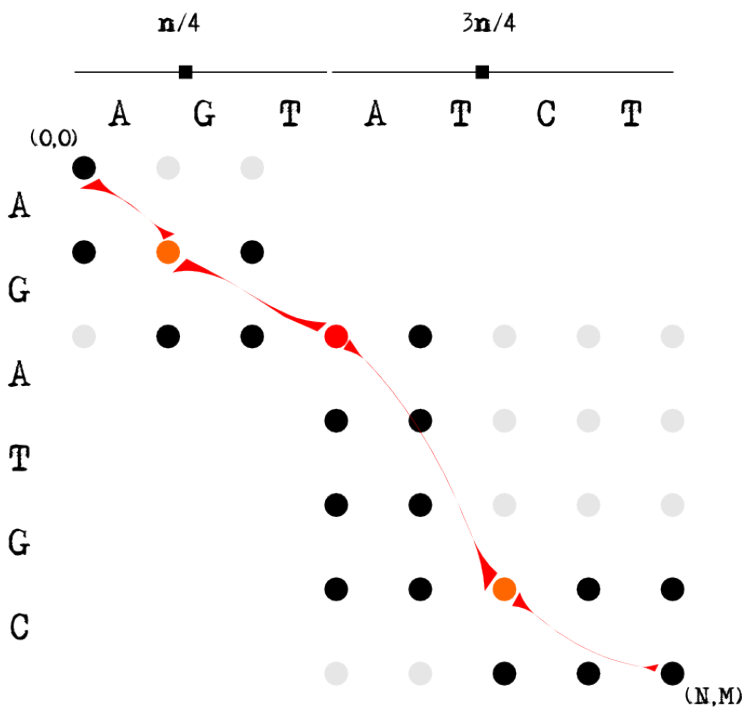
- Heuristique pour comparer une séquence à un ensemble de séquences
  1. Recherche de tous les mots de taille W communs aux séquences avec un score de similarité supérieur à T (*Hit*)
     - par défaut, W = 7 (resp. 3) pour l'ADN (resp. protéines)
     - T = score seuil au-delà duquel la ressemblance entre deux mots n'est pas due au hasard.
  2. Extension des mots trouvés dans les deux directions pour trouver les régions de similitude les plus longues possibles ayant un score supérieur ou égal à un score seuil S
  3. Arrêt de l'extension si
     - Diminution de X du score cumulé par rapport au maximum atteint
     - Score cumulé inférieur à 0
     - Fin d'une des séquences

Optimisation de l'espace

Diviser pour reigner

- Trouver le point de passage de l'alignement optimal pour une colonne donnée
- Réitérer le processus pour chaque point sans perte de complexité

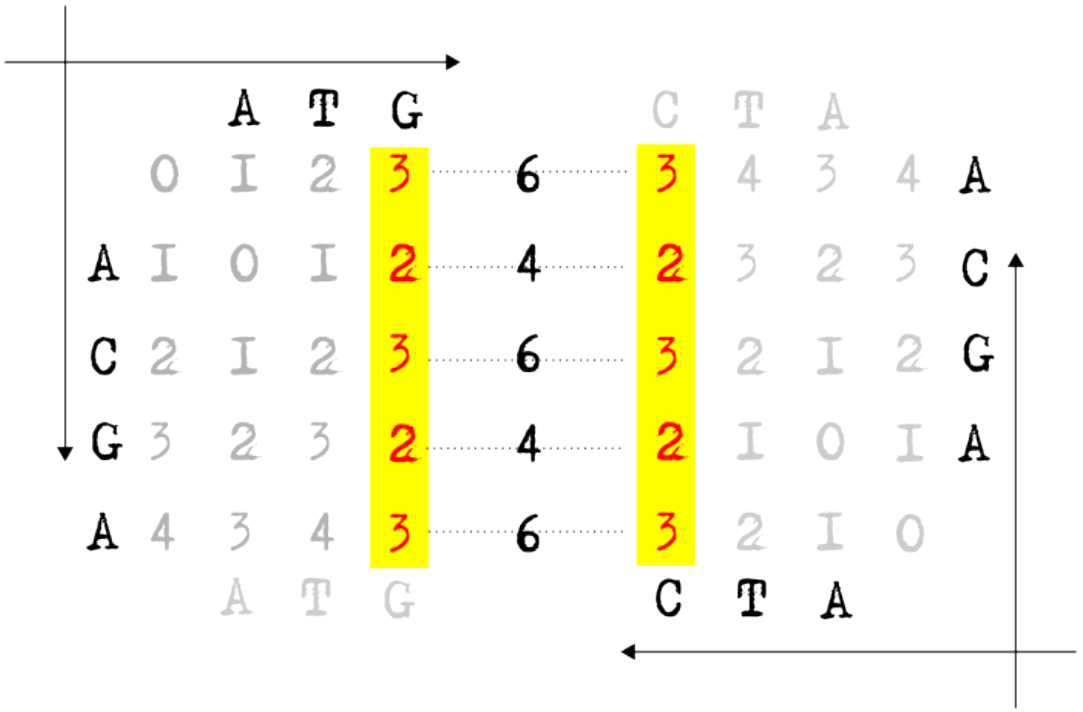




Optimisation de l'espace

Diviser pour reigner

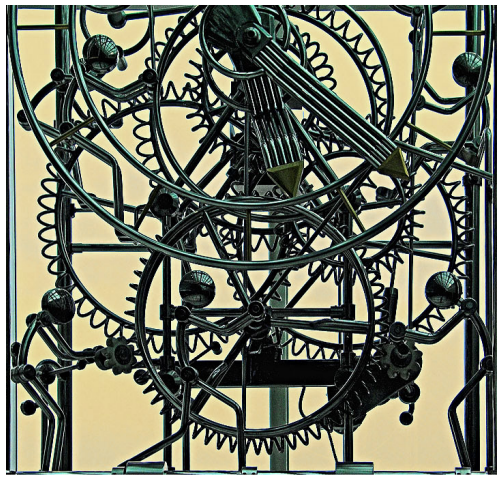
- On effectue le calcul des coûts d'alignement de $S[0..\frac{|S|}{2}]$ et tous les prefixes de T ainsi que ceux de $\overleftarrow{S}[\frac{|S|}{2}+1..|S|]$ et tous les prefixes de $\overleftarrow{T}$ en temps $O(mn)$ et $O(min\{n,m\})$ espace



Optimisation de l'espace

Complexité de l'algorithme

- Pour le calcul du score d'alignement :
  - (Etape 1) – $O(nm)$ en temps, $O(min\{n, m\})$ en espace
- Pour la construction de l'alignement :
  - $O(min\{n, m\})$ en espace
  - Etape 1 – $O(nm)$ en temps
  - Etape 2 – $O(\frac{nm}{2})$ en temps
  - Etape 3 – $O(\frac{nm}{4})$ en temps
  - ...
  - Or $nm + \frac{nm}{2} + \frac{nm}{4} + \ldots = 2nm$, donc en $O(nm)$ en temps



Plan

Bibliographie

Références

1. D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.
2. M.S. Waterman. *Introduction to Computational Biology*. Chapman & Hall, 1995.
3. J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Co, 1997.
4. M. Crochemore, C. Hancart and T. Lecroq. *Algorithmique du texte*. Vuibert, 2001.