Algorithmes fondamentaux pour l'alignement de séquences

Défis de la biologie moléculaire:

Décoder et analyser l'information contenue dans les séquences d'ADN et de protéines.

Une masse de données biologiques. En particulier:

- Environ 200 génomes complètement séquencés et publiés, dont l'homme (23 paires de chro.) et la souris (20 paires de chro.)
- Projets de séquençage de plus de 400 procaryotes et 360 eucaryotes

Intérêt des séquences:

- La séquence nucléotidique d'un gène détermine la séquence d'AA de la protéine
- La séquence d'une protéine détermine sa structure et sa fonction
- Généralement une similarité de séquence implique une similarité de structure et de fonction (l'inverse n'est pas toujours vrai)

Évolution basée, en grande partie, sur la duplication suivie de modification

⇒ beaucoup de redondance dans les banques de données

Les banques de données les plus utilisées:

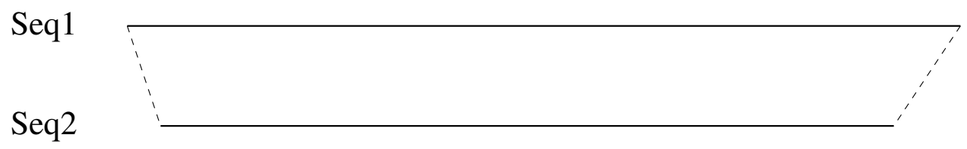
- NCBI - *National Center for Biotechnology Information*
  - GenBank: Séquences d'ADN (3 milliards de paires de bases)
  - Site officiel de BLAST
  - PubMed: Permet la recherche de références
  - COGs: Famille de gènes orthologues
- EMBL - *The European Molecular Biology Laboratory*
- ExPASy - *Expert Protein Analysis System* Protéomique.
  - Swiss-Prot: Séquences de protéines
  - PROSITE: Domaines et familles de protéines
  - SWISS-MODEL: Outil de prédiction 3D de protéines
  - Différents outils de recherche

Problématiques

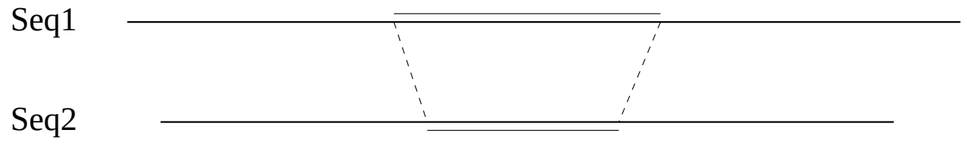
- Est-ce qu'une nouvelle séquence a déja été complètement ou partiellement déposée dans les bases de données?
- Est-ce que cette séquence contient un gène?
- Est-ce que ce gène appartient à une famille connue?
- Quelle est la fonction de cette protéine?
- Existe-t-il d'autres gènes homologues?
- Est-ce que d'autres protéines ont les mêmes domaines ou motifs structuraux?
- Comment ce gène a-t-il évolué par rapport aux autres gènes homologues déja identifiés?
- Existe-t-il des séquences non-codantes semblables (répétitions, régions régulatrices, facteurs de transcription, ARN structuraux)?

Alignement de séquences

Alignement global: Deux séquences de protéines appartenant à la même famille, études phylogénétiques



Alignement local: Séquences de protéines appartenant à des familles différentes, mais domaines communs (sous-unités fonctionnelles conservées)



Recherche de motif:

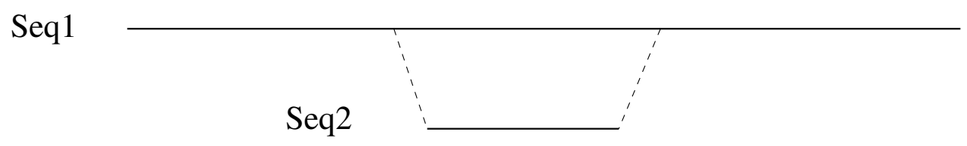
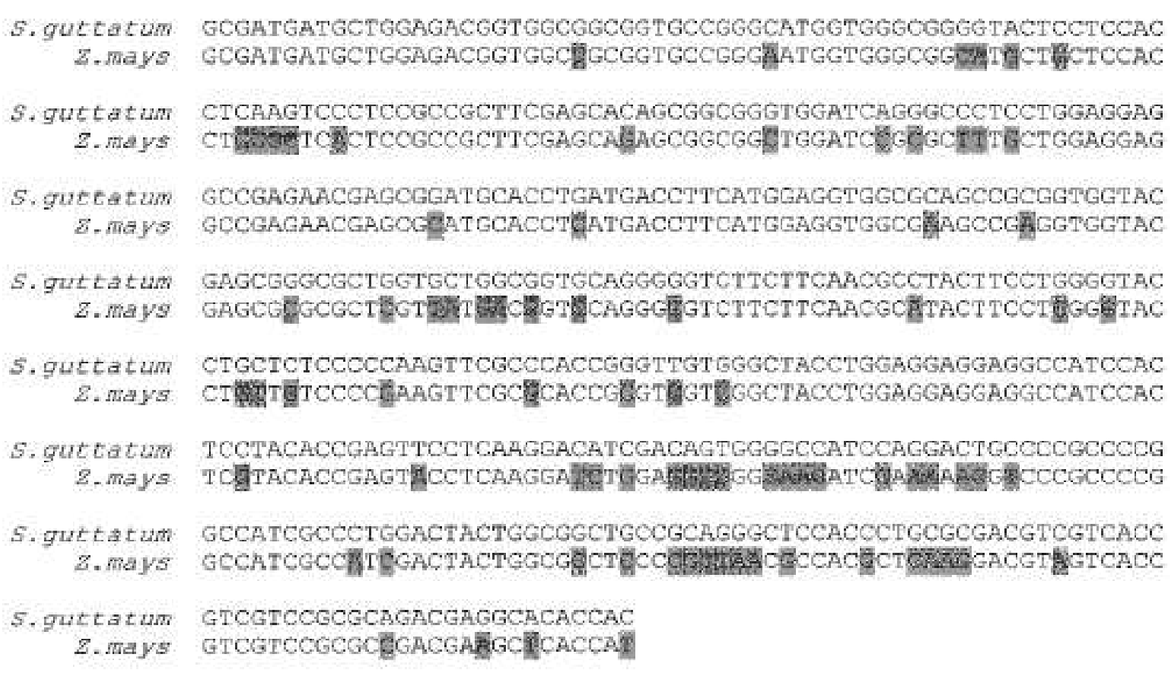


Figure 2A. Sequence comparison of part of the alternative oxidase gene in *Z.mays* and *S.guttatum*



6

Distance d'édition

Pour comparer deux séquences, définir une distance

Distance naturelle: compter le nombre d'insertions, suppressions et substitutions nécessaires pour passer d'une séquence à l'autre.

Exemple: $S_1 = CATAGTG$ et $S_2 = GTCAGGT$.

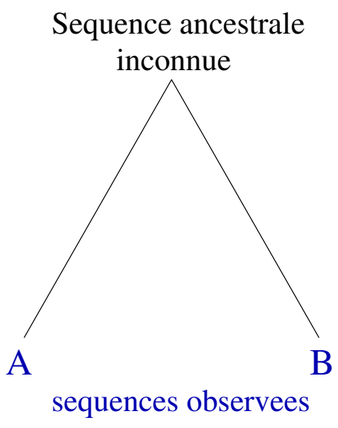
```
S D M I M I M M D  <----  Suite de transformations
C A T   A   G T G        M=match; S=substitution;
G   T C A G G T          I=Insertion; D=suppression
```

Distance d'édition ou de Levenshtein entre S et T: Nombre minimal d'insertions, suppressions et substitutions nécessaire pour transformer S en T (ou réciproquement).

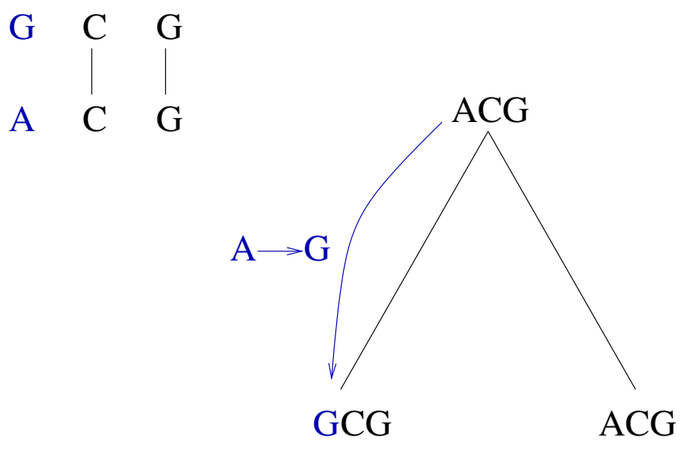
Une insertion/suppression est représentée par un blanc '-'

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

Modèle sous-jacent: Mutations ponctuelles



Exemple: Substitution de caracteres



Programmation dynamique

Pour rédoudre un pb, commencer par résoudre tous les sous-problèmes. Pour ne pas calculer deux fois les mêmes sous-problèmes, conserver les valeurs dans une table

$S = s_1 s_2 \cdots s_m$ de taille m et $T = t_1 t_2 \cdots t_n$ de taille n.

D(i, j): distance d'édition entre $S[1..i]$ et $T[1..j]$. D défini une matrice $(m+1) \times (n+1)$: Matrice de la programmation dynamique.

Exprimer $D(i, j)$ en fonction des valeurs de D pour des paires d'index plus petits que (i, j).

T

D		G	T	C	A	G	G	T
	0	1	2	3	4	5	6	7
C	1							
A	2							
T	3			(i−1,j−1)	(i−1,j)			
A	4			(i,j−1)	(i,j)			
G	5							
T	6							
G	7							D(m,n)

S

Calculer $D(i,j)$ à partir des trois cases $(i-1,j)$, $(i,j-1)$ et $(i-1,j-1)$. En effet, trois alignements possibles:

1. Alignement de $S[1..i-1]$ et $T[1..j]$ suivit de $(s_i,-)$ (suppression).

C A T	A
G T C A	-

2. Alignement de $S[1..i]$ et $T[1..j-1]$ suivit de $(-,t_j)$ (insertion).

C A T A	-
G T C	A

3. Alignement de $S[1..i-1]$ et $T[1..j-1]$ suivit de (s_i,t_j) (substitution ou identité).

C A T	A
G T C	A

Remplissage de la table

Conditions initiales:

$$D(i,0) = i \text{ pour tout } i,\ 0 \leq i \leq m$$
$$D(0,j) = j \text{ pour tout } j,\ 0 \leq j \leq n$$

Relation de récurrence: $i, j > 0$

$$D(i,j) = \min[D(i-1,j)+1, D(i,j-1)+1, D(i-1,j-1)+p(i,j)]$$

$$\begin{aligned} p(i,j) &= 0 \text{ si } s_i = t_j \\ p(i,j) &= 1 \text{ sinon (substitution)} \end{aligned}$$

Complexité: Pour remplir chaque case du tableau, examiner 3 cases. Il y a $O(mn)$ cases. Donc complexité en temps $O(mn)$

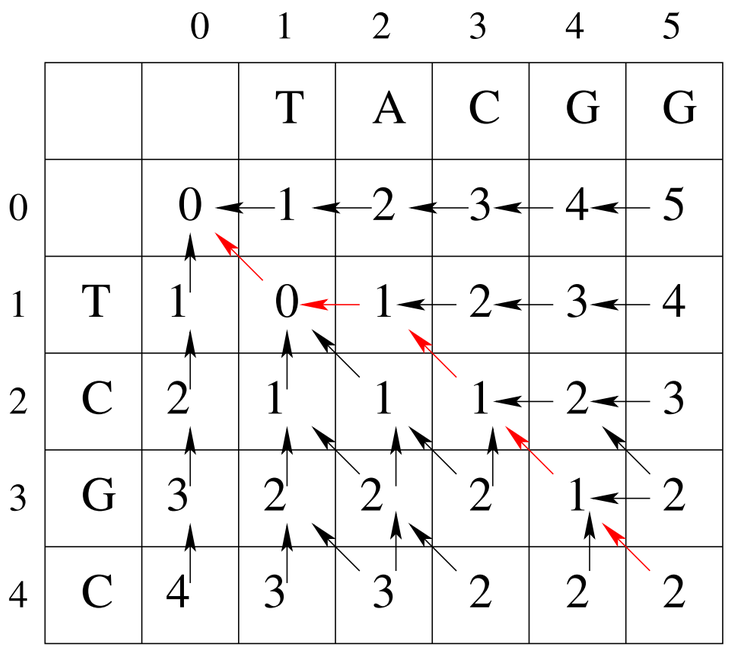
Trouver un alignement optimal

Au cours du remplissage de la table, garder des pointeurs: de $(i-1,j)$ à (i,j) si $D(i,j) = D(i-1,j)+1$; de $(i,j-1)$ à (i,j) si $D(i,j) = D(i,j-1)+1$; de $(i-1,j-1)$ à (i,j) si $D(i,j) = D(i-1,j-1)+p(i,j)$.

Ou, pointeurs implicites: effectuer un test à chaque case.

Un alignement optimal: commencer à la case (m,n) et suivre les pointeurs jusqu'à la case $(0,0)$.

Une case peut contenir plusieurs pointeurs. Plusieurs alignements optimaux possibles.



Alignement résultant:

T - C G C

T A C G G

Distance d'édition avec pondération des opérations

Associer un score à chaque opération

→ d pour insertion/suppression

→ r pour substitution

→ m pour match

$d > 0, r > 0$ et $e \geq 0$. En général $e = 0$

Il faut que $r < 2d$, sinon jamais de substitutions

Relations de récurrence:

$$D(i, 0) = i \times d$$

$$D(0, j) = j \times d$$

$$D(i, j) = \min[D(i, j-1) + d, D(i-1, j) + d, D(i-1, j-1) + p(i, j)]$$

$p(i, j) = e$ si $s_i = t_j$ et $p(i, j) = r$ sinon.

Distance d'édition généralisée

Le score δ dépend des caractères. Par exemple, remplacer une purine par une pyrimidine plus coûteux que remplacer une purine par une purine.

Relations de récurrence:

$$D(i,0) = \sum_{1 \leq k \leq i} \delta(s_k, -), \quad D(0,j) = \sum_{1 \leq k \leq j} \delta(-, t_k)$$

$$D(i,j) = \min[D(i,j-1) + \delta(-,t_j), D(i-1,j) + \delta(s_i,-),$$
$$D(i-1,j-1) + \delta(s_i,t_j)]$$

Si δ est une distance, alors D est une distance

Similarité entre deux séquences

Plutôt que de mesurer la différence entre deux séquences, mesurer leur degré de similarité

$P(a, b)$: Score de l'appariement (a, b). Positif si $a = b$ et ≤ 0 sinon

$V(i, j)$: valeur de l'alignement optimal de $S[1..i]$ et $T[1..j]$.

$$V(i,0) = \sum_{1 \leq k \leq i} P(s_k, -), \quad V(0,j) = \sum_{1 \leq k \leq j} P(-, t_k)$$

$$\begin{aligned} V(i,j) \quad = \quad & \max[V(i,j-1) + P(-,t_j), V(i-1,j) + P(s_i,-), \\ & V(i-1,j-1) + P(s_i,t_j)] \end{aligned}$$

Algorithme de Needleman-Wunch

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X
	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0
		5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1
A	2		6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1
R	-2	6		6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1
N	0	0	2		9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2
D	0	-1	2	4		5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1
C	-2	-4	-4	-5	12		5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1
Q	0	1	1	2	-5	4		6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1
E	0	-1	1	3	-5	2	4		8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1
G	1	-3	0	1	-3	-1	0	5		4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1
H	-1	2	2	1	-3	3	1	-2	6		4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5		5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6		5	0	-2	-1	-1	-1	-1	1	-3	-1	-1
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5		6	-4	-2	-2	1	3	-1	-3	-3	-1
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6		7	-1	-1	-4	-3	-2	-2	-1	-2
F	-4	-4	-4	-6	-4	-5	-5	-5	-2	1	2	-5	0	9		4	1	-3	-2	-2	0	0	0
P	1	0	-1	-1	-3	0	-1	-1	0	-2	-3	-1	-2	-5	6		5	-2	-2	0	-1	-1	0
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2		11	2	-3	-4	-3	-2
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3		7	-1	-3	-2	-1
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		4	-3	-2	-1
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10		4	1	-1
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4		4	-1
B	0	-1	2	3	-4	1	2	0	1	-2	-3	1	-2	-5	-1	0	0	-5	-3	-2	2		-1
Z	0	0	1	3	-5	3	3	-1	2	-2	-3	0	-2	-5	0	0	-1	-6	-4	-2	2	3	
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Matriz **pam250** (*inferior izquierda*) y **BLOSUM62** (*superior derecha*) para pr

Recherche approchée d'un motif

Motif P de taille m, séquence T de taille n, entier k.

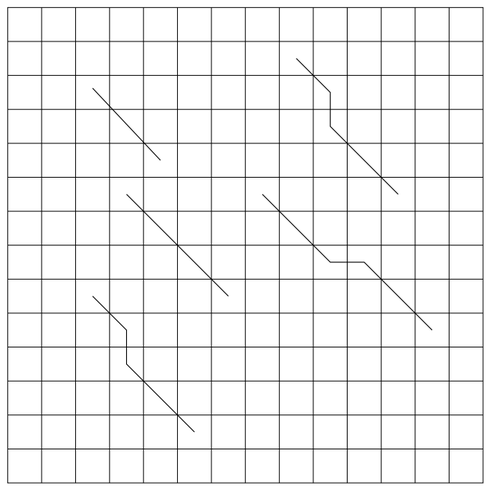
Trouver toutes les occurrences de P dans T à k erreurs près

$D(i,j)$		G	T	C	A	G	G	T
	0	0	0	0	0	0	0	0
C	1							
A	2							
T	3							

- Initialiser la première ligne à 0
- Même relation de récurrence que pour l'alignement de séquences.
- Rechercher à la ligne m toutes les cases contenant des valeurs $\leq k$.
- Suivre les pointeurs jusqu'à la première ligne.

Alignement local - Algorithme de Smith-Waterman

Similarité locale entre deux séquences: Valeur maximale d'un alignement entre deux facteurs des deux séquences.



Exemple: Score de 2 pour match et -1 pour mismatch ou espaces.

```
CAGCAC[TT-GGAT]TCTCGG
  TAGT[TTAGG-T]GGCAT
```

Relations de récurrence:

$$V(i,0) = 0, \quad V(0,j) = 0 \text{ pour tout } i,\ j$$

$$V(i,j) = max[0, V(i-1,j-1) + P(s_i,t_j), V(i-1,j) + P(s_i,-), \\ V(i,j-1) + P(-,t_j)]$$

Réinitialisation de la récurrence (0): Permet d'ignorer un nombre quelconque de caractères en début de séquences.

- Remplire la table
- Rechercher une case c contenant la valeur maximale
- Démarrer à la case c et suivre les pointeurs jusqu'à aboutir à une case de valeur 0.

Variante: Trouver toutes les paires de facteurs dont la valeur de l'alignement optimal dépasse un certain seuil

Considérer les gaps (trous)

Gap: Suite maximale de blancs dans l'une des deux séquences alignées. Possiblement réduit à un seul espace.

```
c t t t a a c - - a - a c
c - - - c a c c c a t - c
```

Contient 4 gaps, 7 espaces, 5 match et 1 mismatch.

Score particulier pour les trous: influence la distribution des espaces d'un alignement optimal.

Quelques motivations biologiques

- Fragments impliqués dans une mutation de longueurs variables (crossing-over, erreurs de réplications, insertion d'ADN par rétro-virus, translocations...)

- Études phylogénétiques: la présence des mêmes trous dans les séquences très significatif

- Deux protéines similaires à beaucoup d'endroits, mais contenant certains domaines différents

- cDNA: ne contient que les exons d'un gène. Retrouver le gène correspondant à un cDNA en l'alignant avec l'ADN

Modèles de pondération

Pondération constante: Score d'un gap indépendant de sa taille;
Constante W_t. De plus, $P(a,-) = P(-,a) = 0$.

Score d'un alignement entre S et T contenant k trous:

$$\sum_{i=1}^{l} P(s_i, t_i) - kW_t$$

Exemple:

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

Score $= 3P(C,C) + 2P(A,A) + P(A,A) + 2P(A,T) - 2W_t$

Pondération affine: Généralisation de la pondération constante.
Modèle de pondération le plus utilisé

W_t: initiation d'un trou

W_e: extension d'un trou

Score d'un trou de taille q: $\omega(q) = W_t + qW_e$

Score d'un alignement de taille l contenant k trous et q espaces:

$$\sum_{i=1}^{l} P(s_i, t_i) - kW_t - qW_e$$

Exemple:

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

Score $= 3P(C,C) + 2P(A,A) + P(A,A) + 2P(A,T) - 2W_t - 5W_e$

Pondération convexe: Chaque espace supplémentaire est moins pénalisé que le précédent

Exemple: Score d'un gap de taille q, $\omega(q) = W_t + \log_e(q)$

Pondération quelconque: Fonction quelconque de la taille du gap.

Recherche d'un alignement optimal - Pondération quelconque

Trois alignements possibles de $S[1..i]$ et $T[1..j]$:

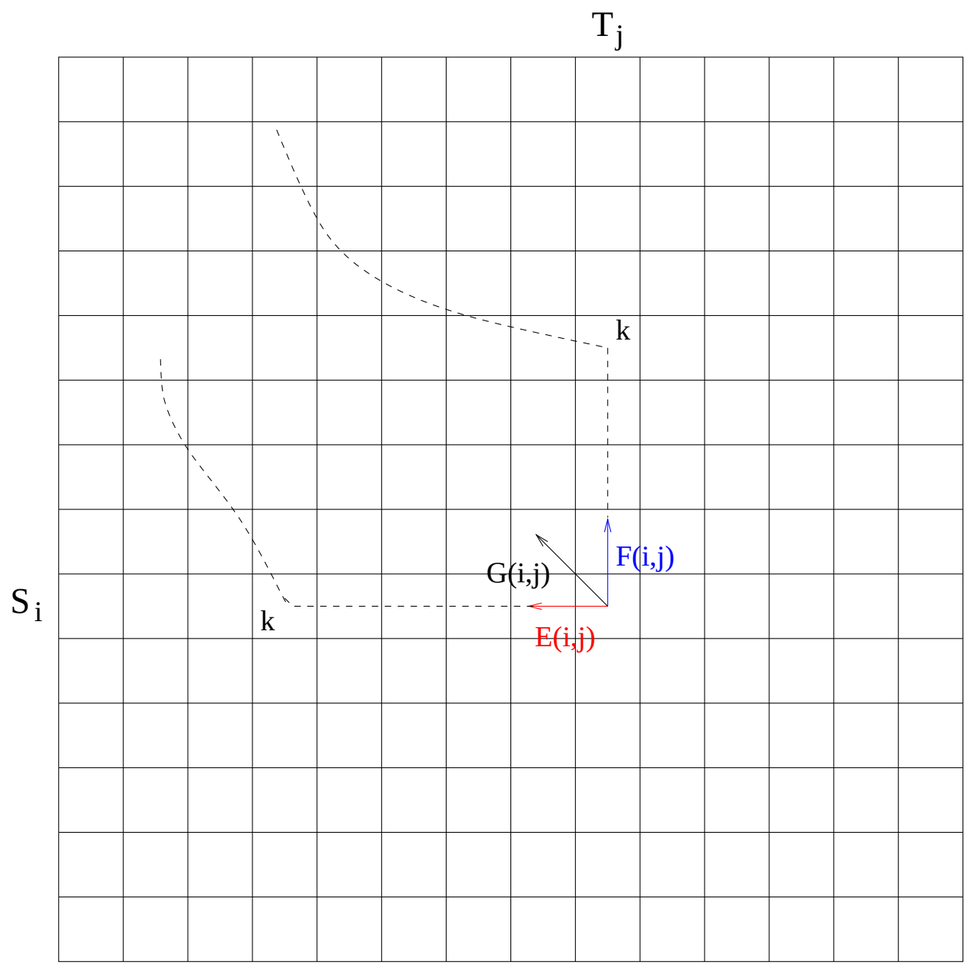
1. Alignement de $S[1..i]$ et $T[1..j-1]$ suivit de $(-, t_j)$.
2. Alignement de $S[1..i-1]$ et $T[1..j]$ suivit de $(s_i, -)$.
3. Alignement de $S[1..i-1]$ et $T[1..j-1]$ suivit de (s_i, t_j).

$E(i,j)$: valeur maximale d'un alignement de type 1.

$F(i,j)$: valeur maximale d'un alignement de type 2.

$G(i,j)$: valeur maximale d'un alignement de type 3.

$V(i,j)$: $\max[E(i,j), F(i,j), G(i,j)]$


T j
S i
k
k
F(i,j)
G(i,j)
E(i,j)

Conditions initiales:

$$V(i,0) = F(i,0) = -\omega(i); \;\; V(0,j) = E(0,j) = -\omega(j)$$

Relations de récurrence:

$$G(i,j) = V(i-1, j-1) + P(s_i, t_j)$$

$$E(i,j) = \max_{0 \le k \le j-1}[V(i,k) - \omega(j-k)]$$

$$F(i,j) = \max_{0 \le k \le i-1}[V(k,j) - \omega(i-k)]$$

Valeur optimale: $V(m,n)$

Variante: Ignore les espaces terminaux. Dans ce cas, $V(i,0) = 0$ et $V(0,j) = 0$. Valeur optimale: une case du tableau contenant la valeur maximale

Complexité: $0(m^2n + mn^2)$

Recherche d'un alignement optimal - Pondération affine

Score d'un gap croît de façon constante après l'initiation du gap

Conditions initiales:

$$V(i,0) = F(i,0) = -W_t - iW_e$$

$$V(0,j) = E(0,j) = -W_t - jW_e$$

Relations de récurrence:

$$V(i,j) = \max[E(i,j), F(i,j), G(i,j)]$$

$$G(i,j) = V(i-1,j-1) + P(s_i, t_j)$$

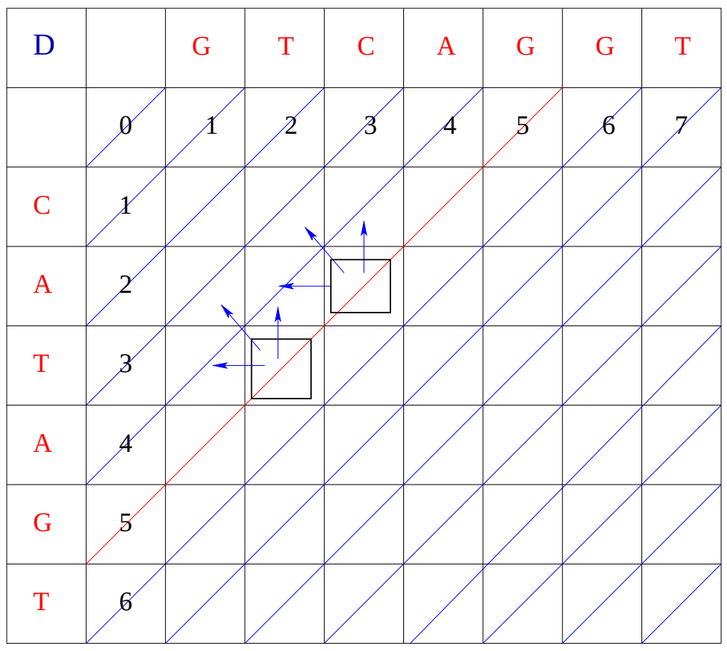
$$E(i,j) = \max[E(i,j-1), V(i,j-1) - W_t] - W_e$$

$$F(i,j) = \max[F(i-1,j), V(i-1,j) - W_t] - W_e$$

Complexité: $O(mn)$

Parallélisme

Comment paralléliser le calcul d'une table de programmation dynamique? Remplire la table par anti-diagonales



Pour chaque anti-diagonale k, on a besoin des anti-diagonales $k-1$ et $k-2$.

Observation clef: Chaque case d'une anti-diagonale k est calculée indépendamment des autres cases de l'anti-diagonale k.

Pour le remplissage d'une anti-diagonale, un processeur peut être assigné au calcul de chaque case.

Complexité: (en temps) $O(n)$