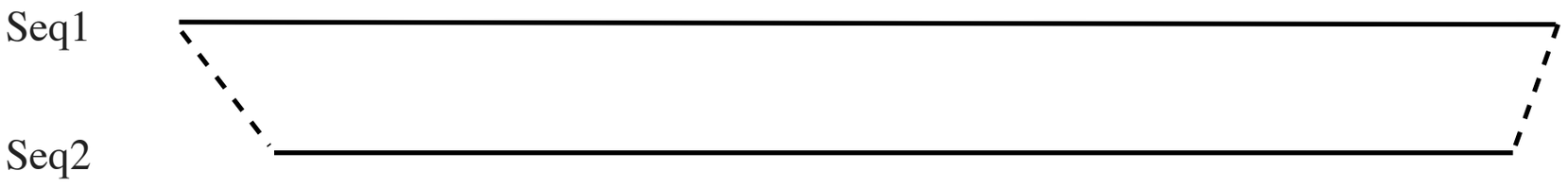
Alignements de séquences

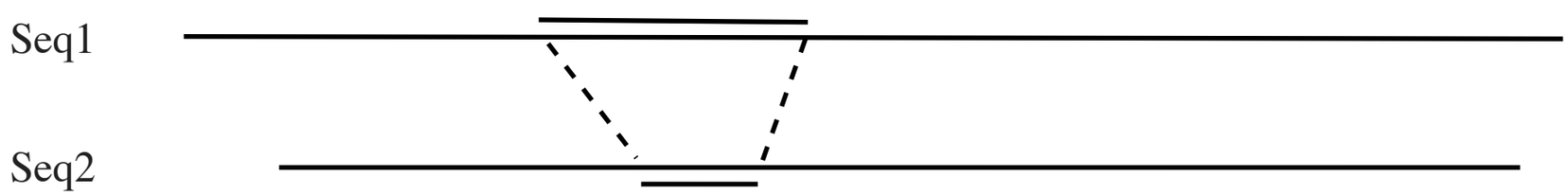
- Problématique
- Distance d'édition + calcul d'un alignement optimal
  - Qu'est-ce qu'une distance
  - Méthode naïve: énumération de tous les alignements possibles
  - Programmation dynamique
  - Alignement global versus local
- Distance versus similarité
- Alignements avec trous

Alignement de séquences

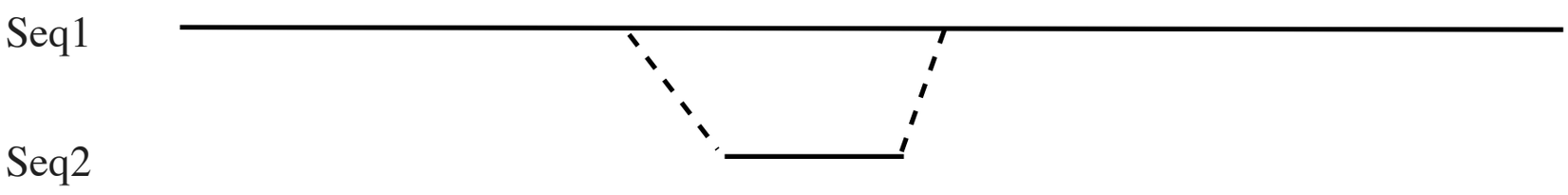
Alignement global: Deux séquences de protéines appartenant à la même famille, études phylogénétiques



Alignement local: Deux séquences de protéines appartenant à des familles différentes mais ayant un ou des domaines communs



Recherche de motif:



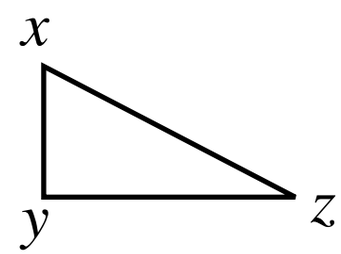
Distance d'édition

Pour comparer des séquences, on va définir une distance:

Définition: Une distance D est une relation ayant les propriétés suivantes:

$$D(x,x) = 0$$

$$D(x,y) = D(y,x)$$



$$D(x,z) \leq D(x,y) + D(y,z) \quad \text{inégalité du triangle}$$

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

Distance d'édition (suite)

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

S	Su	M	I	M	I	M	M	Su
C	A	T		A		G	T	G
G		T	C	A	G	G	T	

Distance d'édition entre S_1 et S_2 : Nombre minimal d'insertions, suppressions et substitutions nécessaire pour transformer S_1 en S_2

Une insertion/suppression est représentée par un tiret '-':

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

Alignement global

Il existe plusieurs alignements possibles étant donné 2 séquences.

Comment trouver un alignement optimal i.e un alignement ayant une distance d'édition minimal??

Idée naïve: Énumérer tous les alignements possibles pour les 2 séquences et en choisir un dont la distance d'édition est minimal.

Exemple: $S_1 = AC$ et $S_2 = AGC$

Les alignements possibles ici sont:

$$\left\{\begin{matrix} AC- & A-C & -AC & AC-- & AC-- & A--C & A-C- & -A-C \\ ACC' & ACC' & ACC' & -ACC' & A-CC' & -ACC' & AC-C' & ACC-' \cdots \end{matrix}\right\}$$

Combien d'alignements?

Si f(n,m) est le nombre d'alignements entre une séquence $a = a_1 \ldots a_n$ de n lettres et une autre $b = b_1 \ldots b_m$ de m lettres alors, on a:

$$f(0,0) = 1$$
$$f(0,m) = 1$$
$$f(n,0) = 1$$

f(n,m) =

Nombre d'alignements de $a_1 \ldots a_{n-1}$ avec $b_1 \ldots b_m$	a_n —
+	
Nombre d'alignements de $a_1 \ldots a_{n-1}$ avec $b_1 \ldots b_{m-1}$	a_n b_m
+	
Nombre d'alignements de $a_1 \ldots a_n$ avec $b_1 \ldots b_{m-1}$	— b_m

$$\Rightarrow f(n,m) = f(n\text{-}1,m) + f(n\text{-}1,m\text{-}1) + f(n,m\text{-}1)$$

Combien d'alignements?

$$\Rightarrow f(n,m) = f(n-1,m) + f(n-1,m-1) + f(n,m-1)$$

	0	1	2	3	4	5
0	1	1	1	1	1	1
1	1	3	5	7	9	11
2	1	5	13	25	41	61
3	1	7	25	63	129	231
4	1	9	41	129	321	681

En fait, on peut montrer que

$$f(n,n) \sim (1+\sqrt{2})^{2n+1} \cdot \sqrt{n}$$

$$\Rightarrow f(1000,1000) \sim 10^{764}$$

Programmation dynamique

Pour résoudre un problème, commencer par résoudre tous les sous-problèmes. Pour ne pas calculer deux fois les mêmes sous-problèmes, conserver les valeurs dans une table m X n

Étant donné 2 séquences, $S = s_1 s_2 \ldots s_m$ et $T = t_1 t_2 \ldots t_n$, on définit

D(i,j): distance d'édition entre le préfixe de taille i de S, $s_1 \ldots s_i$, et le préfixe de taille j de T, $t_1 \ldots t_j$.

D définit une matrice de taille (m+1) X (n+1) qu'on appelle la matrice de programmation dynamique

L'idée est alors d'exprimer D(i,j) en fonction des valeurs de D pour des paires d'indices plus petits que (i,j)

Une matrice de programmation dynamique:

T

S	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)

© notes de cours de Nadia El-Mabrouk

Calculer D(i,j) à partir des 3 cases (i-1,j), (i, j-1) et (i-1,j-1):

1. L'alignement se termine par la suppression de s_i

Alignement optimal de $s_1 \ldots s_{i-1}$ avec $t_1 \ldots t_j$		s_i / —
D(i-1,j)	+	1

2. L'alignement se termine par l'insertion de t_j

Alignement optimal de $s_1 \ldots s_i$ avec $t_1 \ldots t_{j-1}$		— / t_j
D(i,j-1)	+	1

3. L'alignement se termine par l'alignement de s_i avec t_j

Alignement optimal de $s_1 \ldots s_{i-1}$ avec $t_1 \ldots t_{j-1}$		s_i / t_j
D(i-1,j-1)	+	1 si $s_i \neq t_j$
		0 sinon

Remplissage de la table

Conditions initiales: $D(i,0) = i, \quad \forall i \quad 0 \leq i \leq m$

$D(0,j) = j, \quad \forall j \quad 0 \leq j \leq n$

Relation de récurrence pour i, j > 0:

$$D(i,j) = \min \begin{cases} D(i-1,j) & +1 \\ D(i,j-1) & +1 \\ D(i-1,j-1) & +\delta(i,j) \end{cases},$$

Où $\delta(i,j) = 0$ si $x_i = y_j$ et 1 sinon.

Complexité: Pour remplir chaque case de la table, on examine 3 cases.
Il y a $O(nm)$ cases et donc complexité en temps de $O(nm)$

Trouver un alignement optimal

Au cours du remplissage de la table, garder des pointeurs:

- de (i,j) à (i-1,j) si D(i,j) = D(i-1,j) + 1
- de (i,j) à (i,j-1) si D(i,j) = D(i,j-1) + 1
- de (i,j) à (i-1,j-1) si D(i,j) = D(i-1,j-1) + $\delta(i, j)$

Un alignement optimal: Commencer à la case (m,n) et suivre des pointeurs jusqu'à la case (0,0).

Une case peut contenir plusieurs pointeurs: plusieurs alignements optimaux possibles

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

C A T - A G T G -

- G T C A G - G T

Distance versus similarité

Plutôt que de mesurer la différence entre 2 séquences, mesurer leur degré de similarité

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

V(i,j): valeur de l'alignement optimal entre $s_1 \ldots s_i$ et $t_1 \ldots t_j$

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

Relation de récurrence:

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

Algorithme de Needleman-Wunch

Distance d'édition avec pondération des opérations

Associer un score à chaque opération

- d pour une insertion ou suppression $d > 0$
- r pour une substitution $r > 0$
- m pour un match $m \leq 0$ (généralement $m = 0$)

Relations de récurrence:

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

$$D(i,j) = \min \begin{cases} D(i-1,j) & +d \\ D(i,j-1) & +d \\ D(i-1,j-1) & +\delta(i,j) \end{cases}$$

où $\delta(i,j)$ = m si $s_i = t_j$ et r sinon.

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

Distance d'édition généralisée

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

Relations de récurrence:

$$D(0,j) = \sum_{1 \le k \le j} p(-, t_k)$$

$$D(i,0) = \sum_{1 \le k \le i} p(s_k, -)$$

$$D(i,j) = \min \begin{cases} D(i-1,j) & + p(s_i, -) \\ D(i,j-1) & + p(-, t_j) \\ D(i-1,j-1) & + p(s_i, t_j) \end{cases}$$

Recherche approché d'un motif

Problème: On a un "petit" motif P de taille m et une "longue" séquence T de taille n et on veut trouver toutes les occurrences approximatives de P dans T (moins de k erreurs).

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

- initialiser la première ligne à 0
- même relations de récurrence que pour l'alignement global de séquences
- rechercher à la ligne m toutes les cases contenant des valeurs plus petites ou égales à k
- pour trouver un alignement, suivre les pointeurs jusqu'à la première ligne

Recherche de P = ``*CAT*'' à 1 erreur près

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

C A T

. . . C A - . . .

4

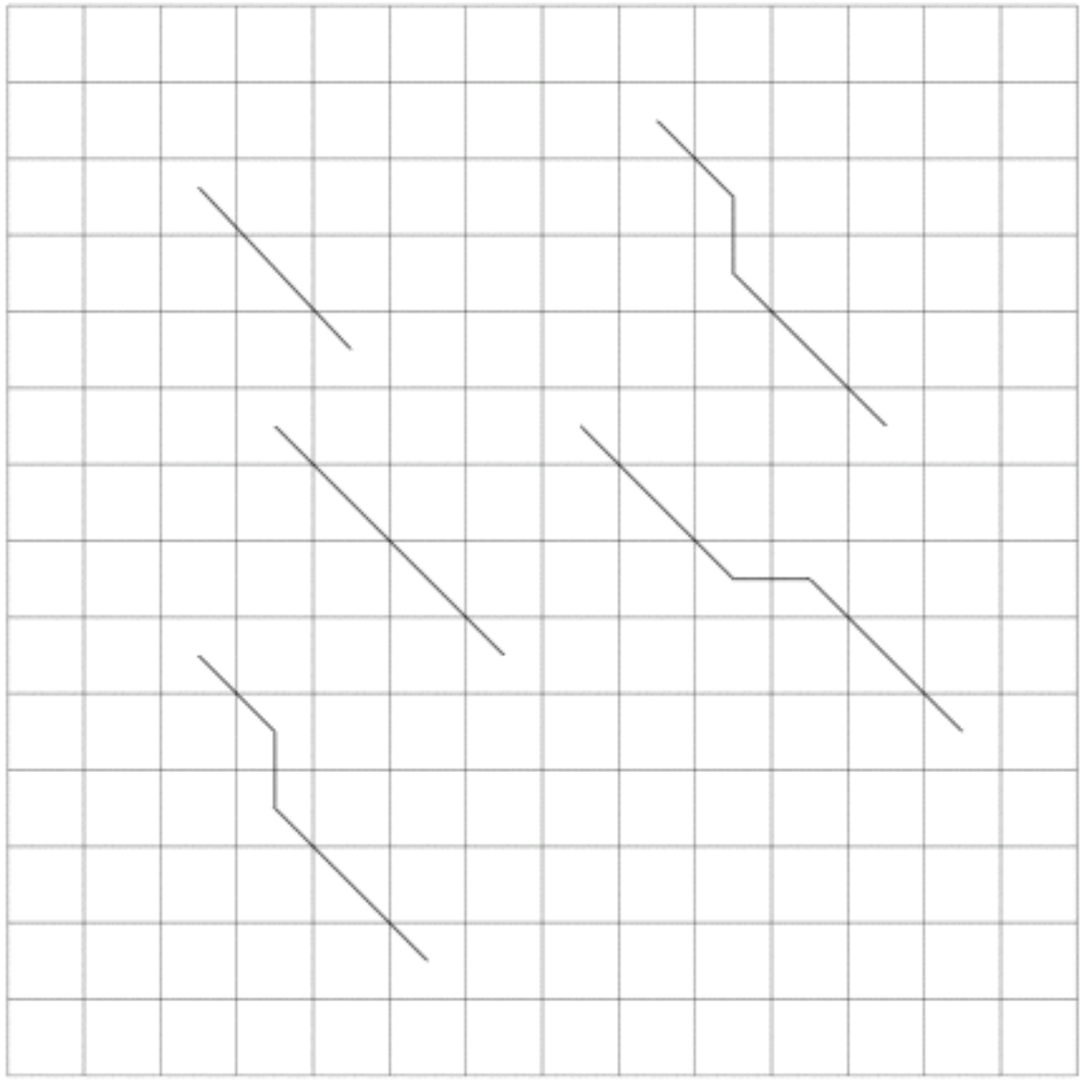
C A T

. . . C A G . . .

5

Alignement local - Algorithme de Smith-Waterman

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



Alignement local - Algorithme de Smith-Waterman

Relations de récurrence:

$$V(0,j) = 0$$
$$V(i,0) = 0$$

$$V(i,j) = \max \begin{cases} 0 \\ V(i-1,j) & +p(s_i,-) \\ V(i,j-1) & +p(-,t_j) \\ V(i-1,j-1) & +p(s_i,t_j) \end{cases}$$

- Le 0 dans la récurrence permet d'ignorer un nombre quelconque de caractères en début de séquence

- Pour trouver un alignement local de score maximal:
  - On remplie la table
  - On recherche une case c contenant la valeur maximale de la table
  - De cette case c, on suit les pointeurs jusqu'à une case contenant la valeur 0

Alignement local

Score 2 pour match, -1 pour mismatch ou insertions/suppressions

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

…G T T A G

G T C A G…

C A G T T …

…C A G G T

C A G - T …

…C A G G T

Considérer les 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.

Modèles de pondération

Pondération constante: Score d'un gap indépendant de sa taille;
Constante W_t.

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
```

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

Modèles de pondération (suite)

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
```

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

Modèles de pondération (suite)

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).

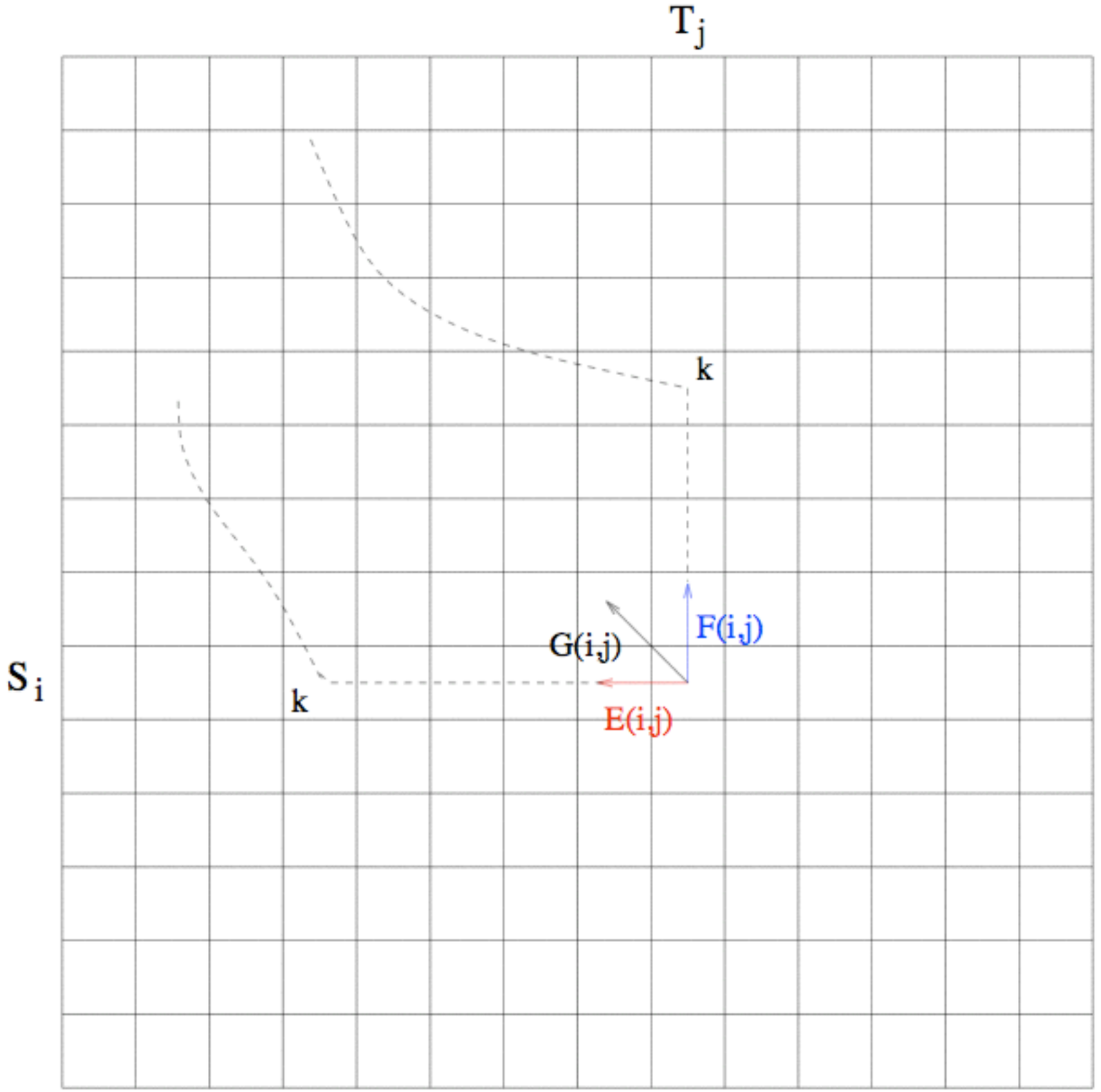
$E(i,j)$: valeur maximale d'un alignement de type 1.

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

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

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

Recherche d'un alignement optimal: pondération quelconque



Recherche d'un alignement optimal: pondération quelconque

Conditions initiales:

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

Relations de récurrence:

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

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

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

Valeur optimale: $V(m,n)$