M2 - STL

Algorithmes sur les séquences en bioinformatique

Cours 2: Algorithmes de comparaison des séquences

Alessandra Carbone
Université Pierre et Marie Curie

Problème : étant données deux (ou plusieurs) séquences d'ADN ou de protéines, trouver le meilleur alignement entre elles.

On commencera avec l'alignement de paires de séquences ou le problème de l'*appariement inexacte*, c-a-d qu'on permettra que certains acides-aminés ne soient pas alignées.

Les algorithmes que l'on introduira sont basés sur la technique de la **programmation dynamique**. Pour chaque algorithme on présent:
- l'intuition qui est à la base du processus récursif
- la définition formelle du processus récursif
- sa complexité

Définition du problème et motivation biologique

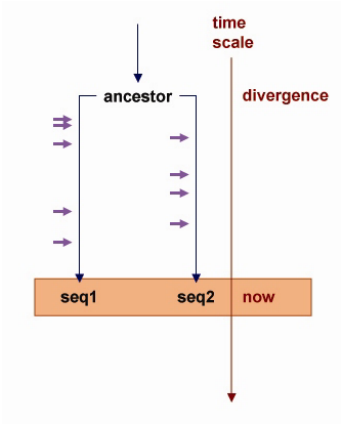
Plusieurs problèmes en bioinformatique demandent une solution du problème de l'alignement entre séquences:

- Reconstruction de séquences longues d'ADN à partir de nombreuses séquences courtes
- Comparer deux ou plusieurs séquences par similarité: recherche de séquences similaires dans une bdd
- Organiser, mémoriser, récupérer et comparer des séquences d'ADN dans les bdd
- Explorer des motifs souvent récurrents dans les séquences nucléotidiques
- Trouver des éléments informatifs dans les protéines (e.g. les domaines fonctionnels) ou les séquences d'ADN (e.g. les sites de regulation)

Homologie de séquences : similarité et différence

Hypothèse théorique. Tout le matériel génétique a un ancêtre commun. La plupart des différences entre familles d'espèces sont dues aux modifications locales entre séquences nucléotidiques:

- insertion – insertion d'une ou plusieurs bases dans une séquence
- délétion – délétion d'une ou plusieurs bases dans la séquence
- substitution – remplacement d'une base par une autre base



Une séquence ACGTACGT a pu évoluer pour donner :

– – ACG– T– A– – –CG– T– – – –
A CACGGTCC TAA TAA TGGCC

– – – AC– GTA –C – –G –T – –
CAG –GA AGA TCTT A GT TC

simulation: après 9500 générations avec des probabilités de
délétion: 0.0001
insertion: 0.001
substitution A/G, T/C: 0.00008
substitution A/C, T/G: 0.00002

Ces deux séquences ont le même ancêtre et on veut connaître leur alignement qui reflète leur origine commune (c'est-à-dire ACGTACGT)

Par exemple, l'alignement pourrait être

–AC AC– GGTCCTAA T– –AATGGCC
CAG –GAA –G–AT– – CTTAGTTC – –

Distance et similarité

Etant données deux séquences, on associe **des poids** aux paires d'acides-aminés/espaces qui représentent le coût des modifications.

La distance entre séquences est la **somme minimale** des poids pour un ensemble de modifications qui transforment l'une dans l'autre.

La similarité entre deux séquences est la **valeur maximale** de la somme des poids.

La distance d'édition

Définition: la distance d'édition entre deux séquences est le **nombre minimale d'opérations** (insertion, délétion et substitution) nécessaires pour transformer une séquence dans l'autre.

Exemple: étant données deux séquences a c c t g a et a g c t a, le nombre minimale d'opérations d'édition nécessaires est 2:

```
a c c t g a          a c c t g a
a g c t g a  substitution  a g c t g a
a g c t a    délétion     a g c t - a
```

L'ordre des modifications n'est pas connu au préalable !

Alignement

Un alignement entre deux séquences S et T est obtenu par insertion d'espaces dans les séquences et a leurs extrémités de telle façon que quand les deux séquences S* et T* sont superposées, un caractère (espace/indel ou lettre) de S* soit comparé à un unique caractère de T* et que les espaces soient opposés aux lettres.

Chaque paire lettre-lettre et lettre-espace (ou espace-lettre) a un **poids σ** associé. D'habitude, l'insertion et la délétion ont le même poids.

poids de l'alignement = $\Sigma_{i=1}^{|S^*|} \sigma(S^*[i],T^*[i])$

Les algorithmes d'alignement (**algorithmes d'optimisation**) recherchent le poids minimale représentant la différence minimale entre séquences (ou bien, ils cherchent la similarité maximale entre S et T).

Exemple 1: L'alignement entre les deux séquences précèdentes

```
– ACAC– GGTCCTA A T– –AATGGCC
  |           |       |  |
CAG –GAA –G–A T– – CTTAGTTC – –
```

Distance 1 – match 0, substitution 1, indel 2 : distance = 31
Distance 2 – match 0, σ(A,T)= σ(G,C)=1, σ(A,G)= σ(A,C)= σ(T,G)= σ(T,C)=1.5, indel 2 : distance = 34
Similarité – match 1, substitution 0, indel -1.5 : similarité = -12.5

Exemple 2:

```
GTAGTACAGCT-CAGTTGGGATCACAGGTTCT
| |  |||      || |   |||  | | | |     | | |  |
GTCGTATTCCTAC - -TTGCGATCAG - AGTTAT
```

Distance 1 – match 0, substitution 1, indel 2 : distance = 18
Distance 2 – match 0, σ(A,T)= σ(G,C)=1, σ(A,G)= σ(A,C)= σ(T,G)= σ(T,C)=1.5, indel 2 : distance = 18
Similarité – match 1, substitution 0, indel -1.5 : similarité = 14

Les probabilités d'insertion, délétion et substitution serons utilisées pour reconstruire l'alignement cherché. On verra dans la suite comment définir une matrice de poids, appelée matrice de substitution. Sur les acides-aminés on a par exemple:

	ALA	ARG	ASN	ASP	CYS	GLN	GLU	GLY	HIS	ILE	LEU	LYS	MET	PHE	PRO	SER	THR	TRP	TYR	VAL
ALA	2	-2	0	0	-2	0	0	1	-1	-1	-2	-1	-1	-4	1	1	1	-6	-3	0
ARG	-2	6	0	-1	-4	1	-1	-3	2	-2	-3	3	0	-4	0	0	-1	2	-4	-2
ASN	0	0	2	2	-4	1	1	0	2	-2	-3	1	-2	-4	-1	1	0	-4	-2	-2
ASP	0	-1	2	4	-5	2	3	1	1	-2	-4	0	-3	-6	-1	0	0	-7	-4	-2
CYS	-2	-4	-4	-5	12	-5	-5	-3	-3	-2	-6	-5	-5	-4	-3	0	-2	-8	0	-2
GLN	0	1	1	2	-5	4	2	-1	3	-2	-2	1	-1	-5	0	-1	-1	-5	-4	-2
GLU	0	-1	1	3	-5	2	4	0	1	-2	-3	0	-2	-5	-1	0	0	-7	-4	-2
GLY	1	-3	0	1	-3	-1	0	5	-2	-3	-4	-2	-3	-5	-1	1	0	-7	-5	-1
HIS	-1	2	2	1	-3	3	1	-2	6	-2	-2	0	-2	-2	0	-1	-1	-3	0	-2
ILE	-1	-2	-2	-2	-2	-2	-2	-3	-2	5	2	-2	2	1	-2	-1	0	-5	-1	4
LEU	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6	-3	4	2	-3	-3	-2	-2	-1	2
LYS	-1	3	1	0	-5	1	0	-2	0	-2	-3	5	0	-5	-1	0	0	-3	-4	-2
MET	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6	0	-2	-2	-1	-4	-2	2
PHE	-4	-4	-4	-6	-4	-5	-5	-5	-2	1	2	-5	0	9	-5	-3	-3	0	7	-1
PRO	1	0	-1	-1	-3	0	-1	-1	0	-2	-3	-1	-2	-5	6	1	0	-6	-5	-1
SER	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2	1	-2	-3	-1
THR	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3	-5	-3	0
TRP	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17	0	-6
TYR	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	-2
VAL	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

Modèles d'alignement

Problème 1: (alignement globale)
Etant données deux séquences S et T ayant à peu près la même longueur, quelle est la similarité maximale entre elles? Trouver le meilleur alignement.

Problème 2: (alignement locale)
Etant données deux séquences S et T, de longueur possiblement différente. Quelle est la similarité maximale entre une sous-séquence de S et une sous-séquence de T? Trouver les sous-séquences les plus similaires.

Problème 3: (globale-locale - alignement libre aux extrémités)
Etant données deux séquences S et T, de longueur possiblement différente. Trouver le meilleur alignement entre sous-séquences de S et de T quand au moins une de ces sous-séquences est un préfix de la séquence originale et une (pas forcement l'autre) est un suffixe.

Définition: un **gap** est l'intervalle d'espaces maximale d'une séquence donnée par rapport à un alignement. La longueur du gap est le nombre d'opérations d'indels effectuées sur la séquence.

Définition: une **fonction de pénalité de gap** est une fonction que mesure le coût d'un gap comme une fonction non linéaire de sa longueur.

Problème 4: (pénalité de gap)
Etant données deux séquences S et T, de longueur possiblement différente. Trouver le meilleur alignement entre les deux séquences utilisant la fonction de pénalité de gap.

Alignement globale optimale en O(nm) en utilisant la programmation dynamique

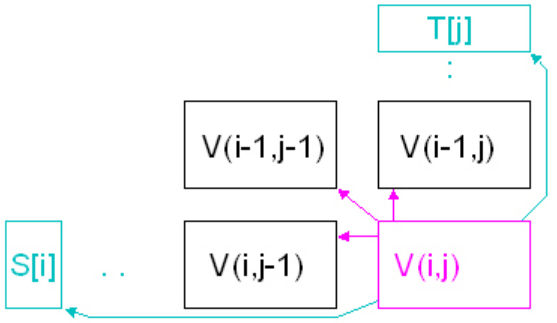
Entrées: deux sequences S, T tels que |S|=n, |T|=m et n~m
Sortie: un alignement optimale et sa valeur

On va resoudre un problème plus compliqué :
V(i,j)= valeur de l'alignement optimale de S[1],…,S[i] avec T[1]…T[j] pour tout $0 \le i \le n$, $0 \le j \le m$

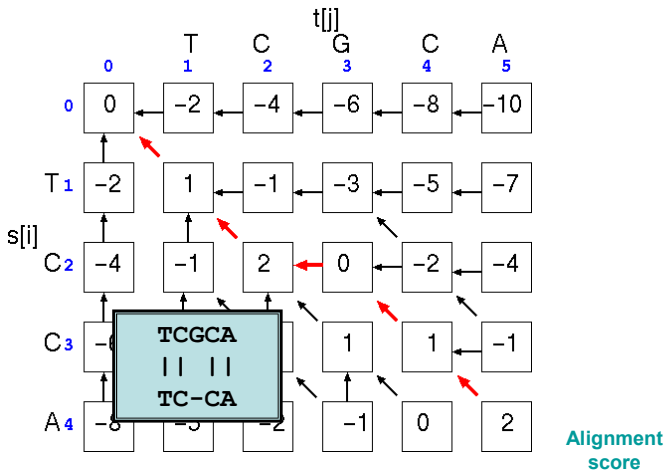
Base: $V(i,0) = \Sigma_{k=0}^{i} \sigma(S[k], -)$
$V(0,j) = \Sigma_{k=0}^{i} \sigma(-, T[k])$

Recurrence:

$$V(i,j) = \max \begin{cases} V(i\text{-}1,j\text{-}1) + \sigma(S[i], T[j]) \\ V(i\text{-}1,j) + \sigma(S[i], -) \\ V(i,j\text{-}1) + \sigma(-, T[j]) \end{cases}$$



Sequence alignment algorithm : Needleman & Wunsch



On peut prouver par induction que cette définition donne bien un alignement optimale.

Exemple: calcul sur tableau

Mismatch = -1
Match = 2

	j	0	1	2	3	4	5	
i			c	a	d	b	d	←T
0		0	-1	-2	-3	-4	-5	
1	a	-1	-1	1				
2	c	-2	1					Time = O(mn)
3	b	-3						
4	c	-4						
5	d	-5						
6	b	-6						
	↑S							

Recherche d'un alignement par backtracking

	j	0	1	2	3	4	5	
i			c	a	d	b	d	←T
0		0	-1	-2	-3	-4	-5	
1	a	-1	-1	1	0	-1	-2	
2	c	-2	1	0	0	-1	-2	
3	b	-3	0	0	-1	2	1	
4	c	-4	-1	-1	-1	1	1	
5	d	-5	-2	-2	1	0	3	
6	b	-6	-3	-3	0	3	2	
	↑S							

Théorème: Le temps de calcul de l'algorithme est O(nm). L'espace de calcul est O(n+m), si seulement V(S,T) est demandé, si non O(nm) pour la reconstruction de l'alignement.

Preuve: devoir.

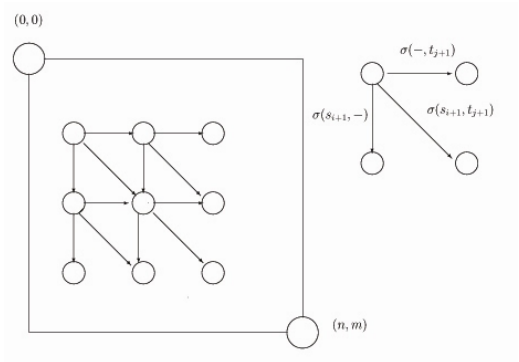
Graphe d'alignement

Il est souvent utile de représenter à l'aide d'un graphe pondéré les solutions obtenues par programmation dynamique de problèmes sur les séquence.

Définition: Soient S et T deux séquences de longueur n et m. Un graphe d'alignement est un graphe dirigé G=(V,E) de (n+1)x(m+1) noeuds, chacun étiqueté par une paire (i,j) distincte ($0 \le i \le n$, $0 \le j \le m$), et avec arêtes pondérés définies comme suit:

1. ((i,j),(i+1,j)) avec poids σ(S[i+1], -)
2. ((i,j),(i,j+1)) avec poids σ(-, T[j+1])
3. ((i,j),(i+1,j+1)) avec poids σ(S[i+1], T[j+1])

Un chemin du noeud (0,0) au noeud (n,m) dans le graphe d'alignement corresponde à un alignement et le poids totale est la valeur de l'alignement. Le but est de trouver le chemin le plus cher de (0,0) à (n,m).



Trois possibilités représentant chacune une étape de l'algorithme.

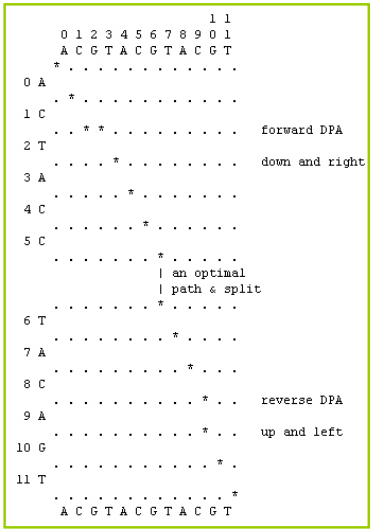
Alignement globale en temps linéaire

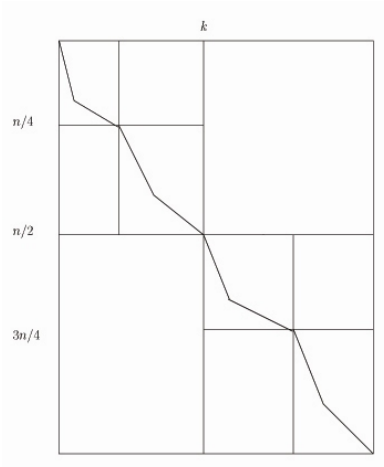
L'algorithme de backtracking appliqué à la recherche d'un chemin, demande la mémorisation de la matrice entière. En conséquence, la complexité en espace est O(nm). Hirschberg a proposé une solution qui donne une complexité d'espace en O(min(n,m)) basée sur la méthode « divide et impera ».

L'algorithme:

Dénoter $V^r(i,j)$ la valeur de l'alignement optimale des derniers i caractères de S contre les derniers j caractères de T.

1. Calculer V(A,B) et sauver en mémoire les valeurs de la n/2-ième ligne, où V(A,B) dénote la *matrice en avant* F.
2. Calculer $V^r(A^r,B^r)$ et sauver en mémoire les valeurs de la n/2-ième ligne, où $V^r(A,B)$ dénote la *matrice en arrière* B.
3. Trouver la colonne k* qui croise le point (n/2,k*) satisfaisant:
   $V(n/2,k^*)+V^r(n/2,m-k^*)=V(n,m)$
4. Partitionner récursivement le problème en deux sous-problèmes:
   (i) trouver le chemin de (0,0) à (n/2,k*)
   (ii) trouver le chemin de (n,m) à (n/2,m-k*)





Lemme: $V(n,m) = \max_{0 \le k \le m} \{V(n/2,k) + V^r(n/2,m-k)\}$

Chaque calcul de programmation dynamique demande la mémorisation de la ligne « n/2 » (la ligne du milieu), mais une fois qu'on trouve le point k* alors cette ligne peut être déchargée. Si n<m on peut mémoriser la colonne du milieu à la place de la ligne.

On en déduit que la complexité d'espace est O(min(n,m)).

Alignement locale

Entrées : deux séquences S et T.
Sortie : la sous-séquence de S et la sous-séquence de T de similarité maximale.

Pourquoi rechercher en locale ?

Pour ignorer les parties non-codantes: Dans l'**ADN**, les régions non-codantes (introns) sont plus sujettes aux mutations que les régions codantes (exons). Quand on cherche un alignement locale entre deux séquences d'ADN, on trouve souvent que le meilleur match concerne des exons.

Pour détecter les domaines des protéines: **protéines** différentes peuvent montrer des similarités locales, les "homeoboxes". Il s'agit dans la plus part des cas de sous-unités fonctionnelles d'une protéine.

Problème de l'alignement locale de suffixes: étant données deux séquences S et T et deux indices i et j, trouver un suffixe (possiblement vide) de S[1…i] et un suffixe (possiblement vide) de T[1…j] tels que la valeur de leur alignement est maximale sur tous les alignements de suffixes de S[1…i] et T[1…j].

La solution du problème de l'alignement locale reviens a trouver la solution maximale du problème de l'alignement locale des suffixes sur tous les indices i et j de S et T.

Terminologie et restriction:

V(i,j) dénote la valeur de l'alignement local des suffixes optimale pour la paire d'indices i, j

Les poids des opérations d'édition sont tels que:

$$\sigma(x,y) = \begin{cases} \geq 0 & \text{si x,y font un match} \\ \leq 0 & \text{si x,y ne font pas de match ou si l'un des deux est un espace} \end{cases}$$

Schema de l'algorithme

1. Calcule l'alignement des suffixes locale (pour tout i et j) de S[1…i] et T[1…j]. Cela est fait en utilisant l'algorithme d'alignement globale.

2. Recherche les résultats et trouve les indices i* et j* de S et T, après lesquels la similarité peut seulement décrémenter.

Définition récursive

Base: pour tout i,j $V(i,0)=0$, $V(0,j)=0$

Recurrence:

$$V(i,j) = \max \begin{cases} 0 \\ V(i-1,j-1)+\sigma(S[1\ldots i],T[1\ldots j]) \\ V(i-1,j)+\sigma(S[1\ldots i],-) \\ V(i,j-1)+\sigma(-,T[1\ldots j]) \end{cases}$$

Calcule i* et j*: $V(i^*,j^*) = \max_{1 \le i \le n,\ 1 \le j \le m} V(i,j)$

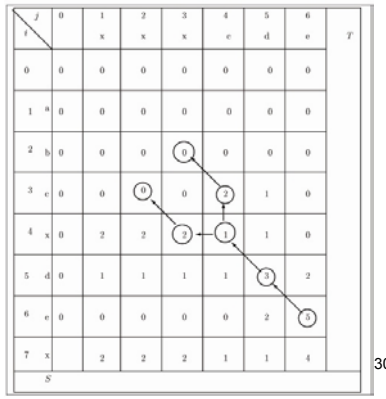
Le 0 dans la base et dans la récurrence permet à l'alignement de commencer n'importe ou dans les séquences, plutôt que le forcer a commencer au début des séquences. La recherche de i* et j* qui donne le meilleur V(i*, j*) permet à l'alignement de terminer n'importe où dans les séquences, plutôt que de terminer à la fin des séquences. Ces deux conditions prises ensemble permettent d'obtenir le meilleur alignement de deux sous-séquences de S et T.

Exemple: σ(match) = 2
σ (mismatch) = -1

Meilleurs sous-sequences:

C- DE
CXDE

XCDE
X -DE



L'alignement locale peut être résolu en :

temps O(nm) : le nombre constante d'operations pas cellule du tableau pour le calcul de V(i,j), donne un temps O(nm) pour le remplissage du tableau. La recherche de V(i*,j*) demande un temps O(nm).

espace linéaire O(n + m) : avec l'algorithme de Hirschberg.

Alignement libre aux extrémités

Entrées: deux séquences S et T, de longueur possiblement différente.
Sortie: le meilleur alignement entre sous-séquences de S et de T quand au moins une de ces sous-séquences est un préfix de la séquence originale et une (pas forcement l'autre) est un suffixe.

Tout opération d'indel aux extrémités des séquences a coût 0. Cette condition permet d'oublier le début et la fin des séquences dans l'alignement et elle permet l'alignement de séquences qui se chevauchent ou quand l'une inclue l'autre.

Exemple: soit S= CACTGTAC et T= GACACTTG
Supposons que σ(match)=2 et que σ(mismatch) = -1.

L'alignement globale optimale a valeur 1 :

```
CAC - - T - GTAC
  | |   |   |
GACACT TG - - -
```

L'alignement locale optimale a valeur 9:

```
 - - CAC- TGTAC
     | ||  ||
GACACTTG - - -
```

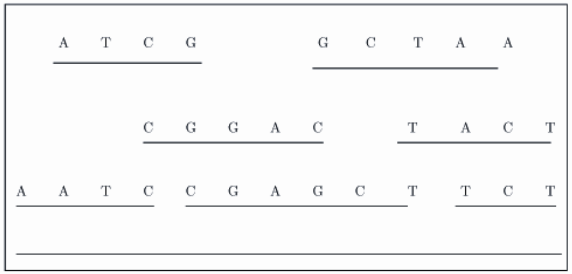
Motivations:

Dans le problème de l'assemblage de séquences shotgun nous avons un grand nombre de séquence qui se chevauchent partiellement et qui proviennent de plusieurs copies de la même séquence d'ADN qui est inconnue et que l'on veut reconstruire. Le problème reviens à utiliser la comparaison de sous-séquences pour déduire la séquence d'origine (en utilisant la partie chevauchante pour coller ensemble les différent buts).

Deux sous-séquences qui proviennent de deux parties différentes de la séquence d'origine aurons valeur d'alignement globale très bas, ainsi que la valeur de l'alignement locale.

Comme deux sous-séquences chevauchantes n'ont pas, avec une forte probabilité, la même position initiale (ni la même position finale) sur la séquence originale, elles aurons une valeur d'alignement globale que sera toujours bas. Par contre, l'alignement libre aux extrémités **sera élevé** parce qu'elles possèdent une partie chevauchante. Le chevauchement sera détecté, et les sous-séquences serons collées ensemble en utilisant l'alignement trouvé.

De façon similaire on traite le cas d'une séquence qui contient une autre séquence.



Algorithme d'alignement à extrémités libres

On fixe les conditions initiales pour permettre poids 0 aux extrémités d'au plus une séquence.

Après avoir rempli la table avec les valeurs V(i,j) on recherche la valeur maximale dans la ligne plus en bas/colonne plus a droite, permettant de terminer au plus une séquence avant l'autre, avec poids zéro pour toutes les opérations d'insertion de gaps de là jusqu'à la fin. La valeur obtenue est la valeur optimale.

La séquence alignée est déterminée à partir de la case (0,0) jusqu'à la fin de la table (ligne plus en bas/colonne plus à droite). Après ce point, toute opération d'introduction de gap jusqu' à (n,m) ne sont pas tenues en compte dans la valeur totale (même si elle sont présentes dans la table).

Base: pour tout i,j $V(i,0)=0$, $V(0,j)=0$

Récurrence: $V(i,j)= \max \begin{cases} V(i-1,j-1)+\sigma(S[1\ldots i],T[1\ldots j]) \\ V(i-1,j)+\sigma(S[1\ldots i],-) \\ V(i,j-1)+\sigma(-,T[1\ldots j]) \end{cases}$

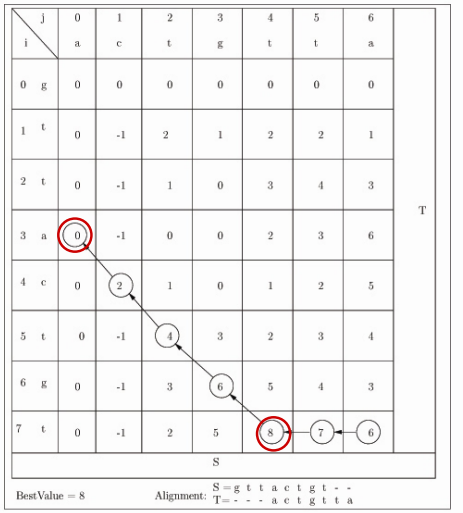
Calcul de i*: $V(i^*,m)= \max_{1 \le i \le n,\, m} V(i,j)$

Calcul de j*: $V(n,j^*)= \max_{n,1 \le j \le m} V(i,j)$

Définition du score d'alignement: $V(S,T)= \max \{V(n,j^*), V(i^*,m)\}$

Complexité en temps:
- trouver la matrice O(nm)
- trouver i* et j* O(n+m)

Complexité en espace:
- calcul de la matrice O(n+m) avec la méthode de Hirschberg
- calcul de i* et j* O(n+m) pour le stockage de la dernière ligne/colonne



Pénalités de gaps

Jusqu'à maintenant les éléments que ont permis la définition du score ont été matches, mismatches et espaces. Un élément important qui est souvent pris en compte est le « gap », cad une suite consécutive maximale d'espaces dans un alignement.

L'idée est de traiter le gap comme un élément unique plutôt que de lui associer une valeur qui est la somme d'espaces. Il y a plusieurs façon de traiter la valeur d'un gap et on présentera plusieurs modèles.

#gaps= nombre de gaps dans un alignement

Exemple: l'alignement suivant à 8 gaps et un totale de 11 espaces.

– ACAC– GGTCCTA A T– –AATGGCC
| | | | |
CAG –GAA –G–A T– – CTTAGTTC – –

Motivations

L'insertion ou la délétion de sous-séquences entières a lieu souvent comme **un seul événement mutationnel**. En plus, plusieurs de ces événements mutationnels peuvent créer des gaps de tailles différentes. Pour éviter d'associer des valeurs qui sont trop importants à ces événements mutationnels il faut que l'on traite les gaps comme des unités en soit même.

Exemple: considérons l'alignement d'ARN messagers (qui codifient une protéine, sans contenir des traces d'introns) avec des séquences d'ADN qui contiennent exons et introns. On sait que l'alignement contera plusieurs gaps possiblement très longs correspondant aux introns. L'utilisation d'**une bonne pénalité de gap nous permettra d'éviter des scores très faibles** pour ces alignements et donc nous permettra de trouver l'alignement vrai.

Modèle à pénalité de gap constante : espace n'a pas de poids
gap a poids W_g indépendant de sa longueur

Le problème reviens a maximiser $\Sigma\, \sigma(S'[i], T'[i]) + W_g \times \#gaps$

ou S' et T' sont les sequences S et T avec espaces.

Modèle à pénalité de gap affine : le gap a 2 poids
W_g ouverture du gap
W_s extension du gap avec un espace
avec un poids totale $W_{totale} = W_g + qW_s$
pour un gap de longueur q

Le problème reviens a maximiser $\Sigma\, \sigma(S'[i], T'[i]) + Wg \times \#gaps + W_s \times \#espaces$

Algorithme avec pénalité de gap affine

Pour aligner S et T, on considère les préfixes S[1…i] et T[1…j]. Un alignement des deux préfixes rentre dans l'une des catégories suivantes:

1. S ---------- i
   T ---------- j

où les caractères S[i] et T[j] sont alignées entre eux.
Cette hypothèse inclus les cas S[i]=T[j] et S[i] ≠T[j].

2. S ---------- i_____
   T ---------- j

où le caractère S[i] est alignée à un caractère strictement à la gauche de T[j].
L'alignement termine alors avec un gap sur la droite dans S.

3. S ---------- i
   T ----------j_____

où le caractère T[j] est alignée à un caractère strictement à la gauche de S[i].
L'alignement termine alors avec un gap sur la droite dans T.

Notation:

G(i,j)	la valeur maximale des alignements de type 1
E(i,j)	la valeur maximale des alignements de type 2
F(i,j)	la valeur maximale des alignements de type 3
V(i,j)	la valeur maximale de l'alignement

Comme dans les autres algorithmes d'alignement on donne une définition récursive de la table d'alignement. Il y a 3 relations de récurrence a définir, pour G(i,j), E(i,j) et F(i,j).

Exemple: on considère E(i,j). On doit regarder les opérations d'insertion d'un espace et assigner la bonne valeur : cela dépends du poids des espaces (qW_s), mais aussi du poids de l'ouverture d'un gap (W_g). Elle sera calculée en regardant les valeurs de la table calculées aux récursions précédentes. Il y a deux cas possibles:

1. S termine à la gauche de T, comme pour l'étape courante. Nous ajoutons un poids d'extension à la valeur précédente: $E(i,j-1)+W_s$.

2. S et T terminent au même endroit (type 1). Nous ajoutons un poids d'ouverture et un poids d'extension: $V(i,j-1)+W_g+W_s$.

 Le max entre les deux donne la valeur de E(i,j).

Définition récursive

Base

$V(0,0) = 0$
$V(i,0) = E(i,0) = W_g+iW_s$
$V(0,j) = F(0,j) = W_g+jW_s$

Recurrence

$V(i,j) = \max \{ E(i,j),F(i,j),G(i,j) \}$
$G(i,j) = V(i-1,j-1)+ \sigma(S[i],T[j])$
$E(i,j) = \max \{E(i,j-1)+W_s, V(i,j-1)+W_g+W_s\}$
$F(i,j) = \max \{F(i-1,j)+W_s, V(i-1,j)+W_g+W_s\}$

Complexité en temps: O(nm) parce qu'on calcule 4 matrices à la place d'une seule.
Complexité en espace: O(nm) pour une implémentation de base.

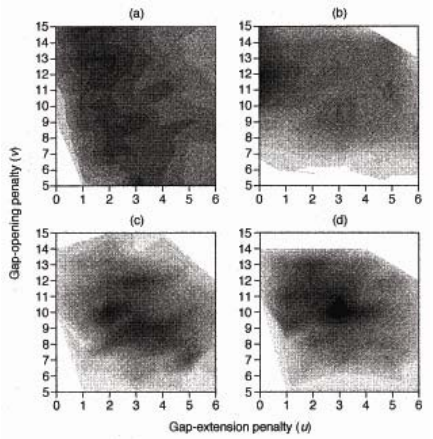
Modèle à pénalité de gap convexe : chaque espace ajouté à un gap contribue moins au poids que le précèdent. On utilise par exemple la fonction de poids $W_g \cdot \log(q)$, ou q est la longueur du gap.

Le problème est résoluble en temps O(nmlog(m)).

Modèle à pénalité de gap arbitraire : n'importe quelle fonction de poids est acceptable, et le poids est fonction de la longueur q du gap.

Le problème est résoluble en temps O(nm(m+n)).

Distribution de meilleures valeurs de gap pour des programmes d'alignement différents



(a) Pairwise Alignment
(b) ClustalW
(c) Randomized Iterative Method
(d) Double Nested Iterative Method

Profiles calculés sur 54 familles, ou les methodes ont pu rejoindre le 99% de paires correctement alignées.

En 3eme dimension: pourcentage de paires correctement alignées

Why « pairwise alignment » again ?

Detection of homologous proteins

Genomes with the 60% of genes of unknown function
Ex: *Plasmodium falciparum*

Detection of protein binding sites

Larger families of homologous proteins allow to study a broader variety of evolutionary signals.

Sensitive multiple alignement

Good pairwise alignment is demanded to avoid error propagation especially for multiple alignment of sequences with low homogy

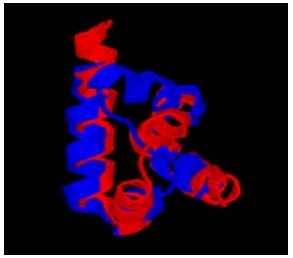
We look for very weak signals

An example: looking for proteins with similar structure

protéine ribosomale L20
Aquifex aeolicus

protéine
polyA binding Homo sapiens

Superposition of the two structures



bacteria and human

Structural alignment program PROSUP

```
bacteria: |----WIARINAAVRA--YGLNYSTFINGLKKAGIELDRKILADMAVRDPQAFEQVVNKVKEALQVQ-
  equiv.: |    **********    *******           ************************
   human: |HRQALGERLYPRVQAMQPAFASKITG------MLLELSPAQLLLLLASEDSLRARVDEAMELIIAHG
Identity: |       #    # #                                          #    #
```

Number of identities = 5; *RMSD* = 2.41Å

Sequence alignment programs fail

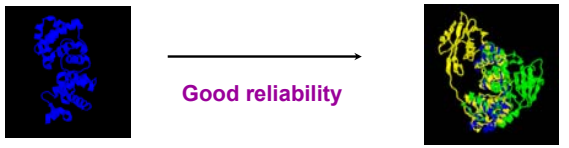
Sequence alignment program CLUSTALW

```
bacteria: |~WIARINAAVRAYGLNYSTFINGLKKAGIELDRKILADMAVRDPQAFEQVVNKVKEALQVQ
   human: |HRQALGERLYPRVQAMQPAFASKITGMLLELSPAQLLLLLASEDSLRARVDEAMELIIAHG
Identity: |   #               #         ##    #             #
```

Number of identities = 6

Alignment and Homology Search Reliability

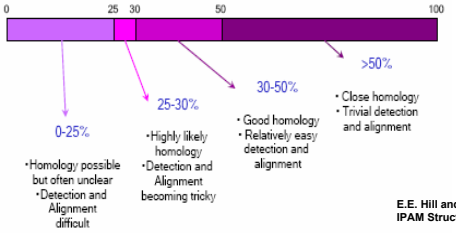
A) **Structure**



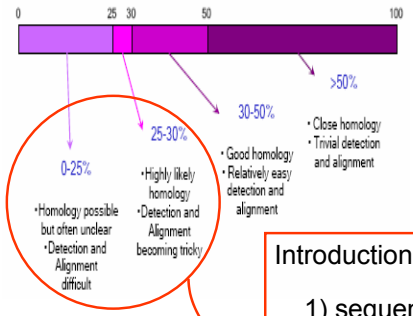
B) **Sequence**

MFPDAHCELVHRNPFELLIAVVLSAQ

MFPDAHCEL--VHRNPFELLIAVVLSAQ
PGLRLPGC-----VDAFEQGVRAILGQL
YHDNEWGVPETDSKKLFEMICLEGQQAG



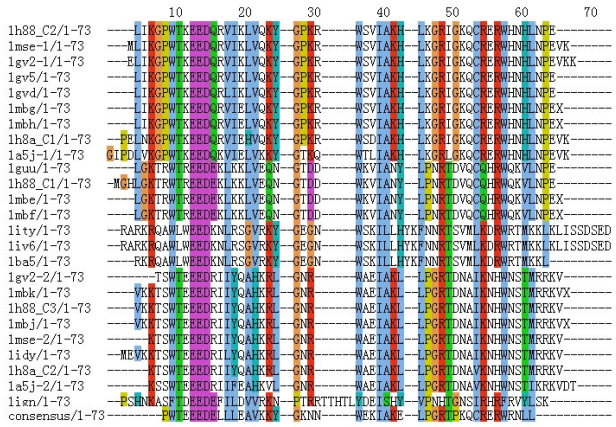
E.E. Hill and S. E. Brenner, IPAM Structural Proteomics, 2004



Introduction of structural information in

1) sequences

2) alignment parameters:
- substitution matrices
  (Overington et al., 1992, Teodorescu et al., 2004)
- gap penalties
  (Lesk et al., 1986)



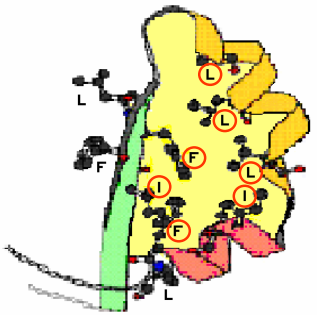
Physico-chemical properties of amino-acids:

ED	ionizable	TQS	H-bonding
ILVAWMCF	non-polar/hydrophobic	HY	aromatic
RK	polar/hydrophilic	P	cyclic/proline

For highly divergent proteins :

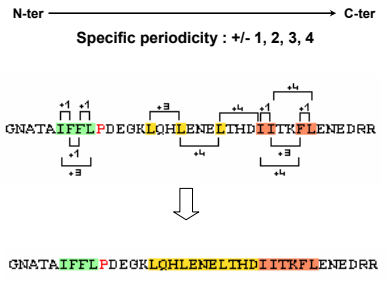
Hydrophobic amino-acids are the only ones conserved in protein families after strong evolutionary pressure

Hydrophobic Blocks



Hydrophobic core	⇨	**Folding stability**
Surface	⇨	**Protein interactions**

Prediction of Hydrophobic Blocks in Sequences

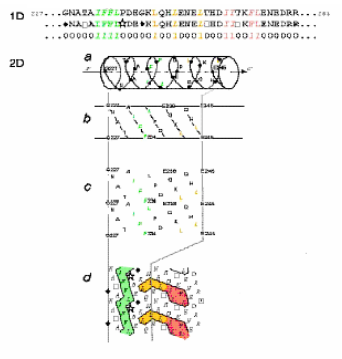


Specific periodicity in sequences :

1. α–helices on protein surfaces: contacts at distance 3-4
2. β-sheets n protein surfaces: contacts at distance 2
3. Secondary structures in the hydrophobic core display chains of aa at distance 4

Hydrophobic Cluster Analysis (Mornon et al.)

α-helical 2D representation



+ manual alignment

Gaboriaud et al., 1987

Test of several evolutionary hypothesis on protein families with low homology

145 protein families : 613 sequences (< 30 % pairwise identity)
from the HOMSTRAD database ;
structural alignment with FUGUE of 1426 protein pairs

Statistics of hydrophobic blocks
Substitution matrices
Gap treatment inside and outside blocks
An alignment method based on hydrophobic blocks

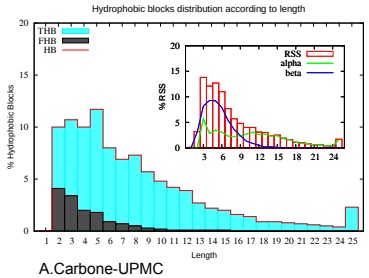
Statistics on Hydrophobic Blocks

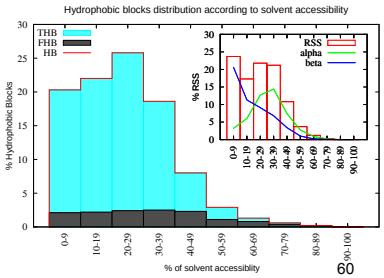
% RSS overlapping HB :	**% HB overlapping RSS** :
89.8%	85.7% ⇨ THC

Ferritin (1RYT)

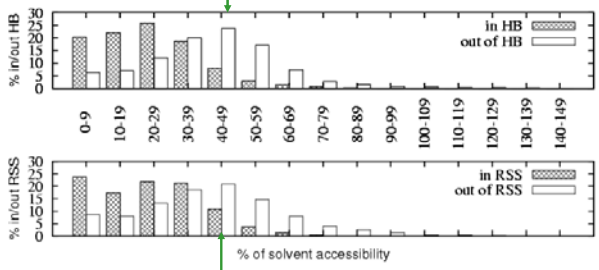
Regular Secondary Structures (RSS) | Hydrophobic Blocks overlapping RSS | Hydrophobic blocks

	RSS	**HB**	**THB**	**FHB**
Mean length :	8.4	8.1	8.8	4.2
Mean % solvent accessible surface	25.3	24.4	23.7	33.2





40.6%
average solv acc outside Hydrophobic Blocks



40.5%
average solv acc outside RSS

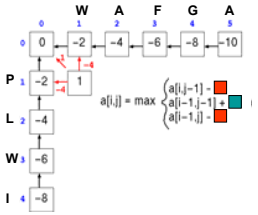
How to introduce the hydrophobic blocks signal predicted from sequences in the alignment algorithm ?

Alignment of sequences using Hydrophobic Blocks

Needleman & Wunsch

All residues are under the same evolutionary pressure

Substitution Matrix
Gap penalties

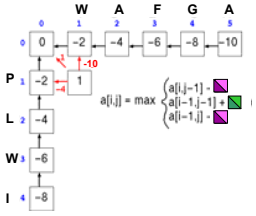


PHYBAL

Evolutionary pressure
inHBlocks > outHBlocks

inHBlocks → Substitution Matrix, Gap penalties

outHBlocks → Substitution Matrix, Gap penalties



Needleman and Wunsch, 1970

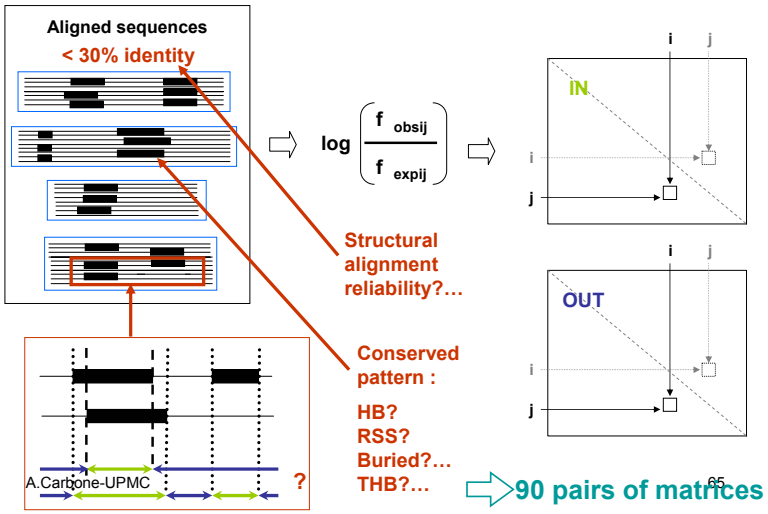
Baussand, Deremble, Carbone, *Proteins*, 2006

Substitution matrices construction and evaluation of the system

How to construct matrices specific to the hydrophobic block context ?

How to evaluate the system to observe alignment accuracy improvement ?

Substitution matrices construction hypothesis

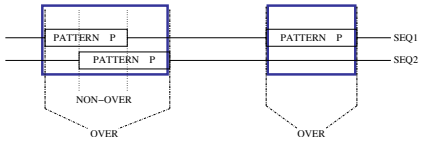


90 matrices coming from…

A) 3 structural alignements
- FUGUE sequence information+ substitution matrix based on structural information
- PROSUP, smallest RMSD
- PROSUP, largest number of equivalents, i.e. residues which are <5Å apart

B) 5 structures: RSS buried residues (<30% solv acc)
HB THB overlapping THB and RSS

C) 2 ways to overlap structural regions

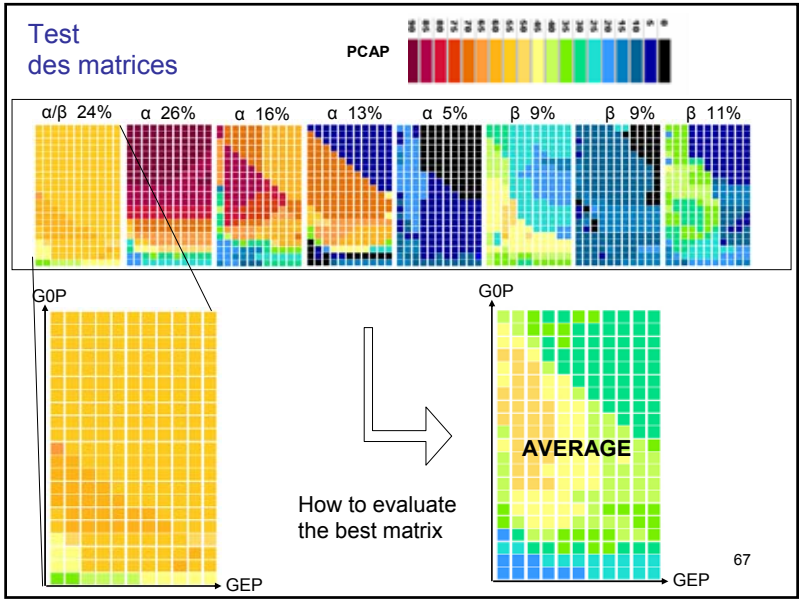


D) 3 equations to define scores of substitution

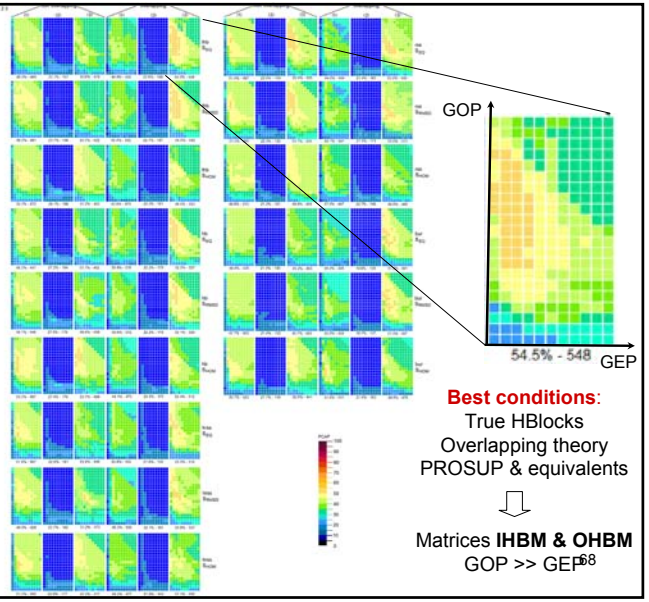
$$\log\left(\frac{f^{pat}_{obsij}}{f^{seq}_{expij}}\right)$$

…and 2 more

Test des matrices



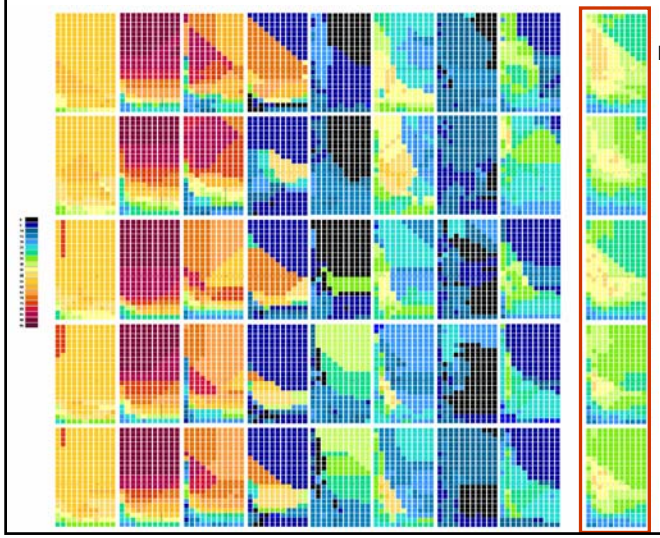
Evaluation :
Average PCAP on different combinations of GAP values for the 8 couples of proteins with low homology



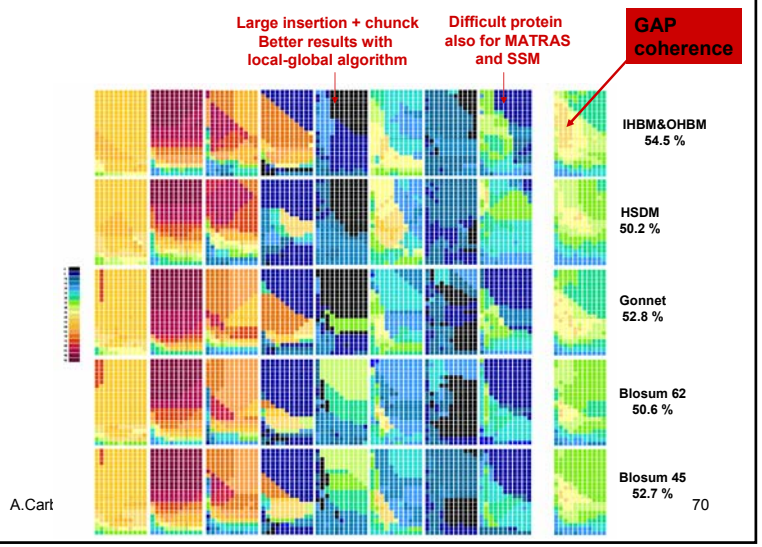
Best conditions:
True HBlocks
Overlapping theory
PROSUP & equivalents

Matrices **IHBM & OHBM**
GOP >> GEP

Comparison to other substitution matrices

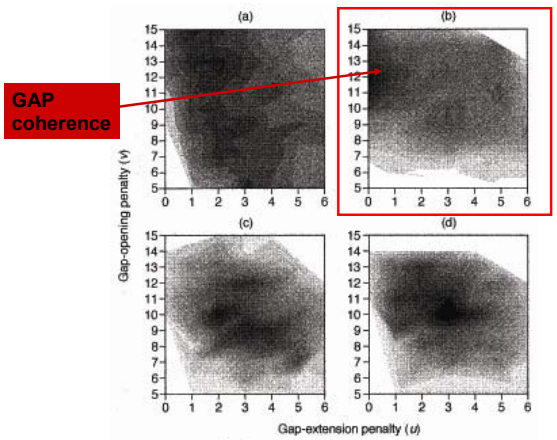


Landscape of PCAP according to gap penalties



A.Cart

Best gap values distribution for different alignment programs (Osamu Gotoh, JMB, 1996)



(a) Pairwise Alignment
(b) ClustalW
(c) Randomized Iterative Method
(d) Double Nested Iterative Method

Profiles computed on 54 families, where the methodes reached 99% PCAP

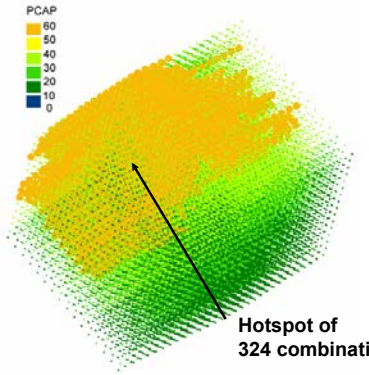
3rd dimension: percentage of correctly aligned pairs (PCAP)

A.Carbone-UPMC

Searching for optimized gaps in 4 dimensions

In HB parameters:
IHBM
bGOP: 0 to 20
bGEP: 0 to 10

Out HB parameters:
OHBM
GOP: 0 to 20
GEP: 0 to 10



Better PCAPs and larger consistent zones

α/β 24%	α 26%	α 16%	α 13%	α 5%	β 9%	β 9%	β 11%
+7.5	+3.9	=	+9.2	+11.9	+5.5	+5.1	+4.8

A.Carbone-UPMC

Comparison with Needleman&Wunsch

PHYBAL

IHBM and OHBM

GOP/GEP : 0 to 20/10
bGOP = GOP and bGEP = GEP

2 dimensional analysis : 231 combinations
4 dimensional analysis : 324 combinations on hotspot

Needlman & Wunsch

HSDM
Blosum30
Blosum62
Gonnet

GOP/GEP : 0 to 20/10 (231 combinations)

Global stability : (comparison with other systems) number of PCAPs which are at most 5% away from the best PCAP obtained on all systems

Local stability : (robustness of a system) number of PCAPs which are at most 2% away from the highest of the system

Domingues set

(Domingues F.S *et al.*, 2000)

165 proteins

127 protein pairs with <30% sequence identity and with a reference alignments performed by ProSup :

- at least 35 equivalent residues (i.e. residues which are <5A apart)

- share all secondary structural elements in the hydrophobic core

Method	PCAP (CAP)	glo	loc
PHYBAL hot-spot	**43.7 (6461)**	324	110
PHYBAL-2D + IHBM & OHBM	42.6 (6268)	49	21
Gonnet	40.2 (5982)	17	22
HSDM	43.5 (6454)	76	21
Blosum 62	41.4 (6172)	27	19
Blosum 45	41.6 (6259)	34	23
Blosum 30	38.1 (5694)	0	18
PAM250	39.3 (5797)	2	17
Johnson	41.5 (6051)	42	32
Remote Homo	32.5 (4783)	0	61

BAliBASE (Thompson J. *et al.*, 1999)

181 proteins pairs with <25% sequence identity and ranked in 6 classes of % identity

Comparison of predicted alignments is carried on **core blocks** only, i.e. regions that can be reliably aligned.

Method	≤ 12	glo	loc	12-15	glo	loc	15-17	glo	loc
PHYBAL hot-spot	**43.2 (296)**	10	3	**39.0 (1203)**	11	3	**40.1 (1289)**	80	21
PHYBAL-2D + IHBM & OHBM	41.6 (244)	9	4	37.3 (1018)	5	4	37.7 (1211)	9	4
Gonnet	37.8 (246)	0	3	29.8 (816)	0	20	35.5 (1145)	8	2
HSDM	24.4 (176)	0	4	28.2 (823)	0	7	29.5 (1100)	0	5
Blosum 62	33.0 (239)	0	1	31.2 (964)	0	6	34.0 (1260)	0	12
Blosum 45	36.3 (243)	0	7	31.4 (898)	0	3	33.7 (1207)	0	9
Blosum 30	26.8 (197)	0	3	28.4 (830)	0	4	32.0 (1179)	0	5
PAM250	22.4 (174)	0	9	26.3 (821)	0	20	27.4 (905)	0	8
Johnson	24.8 (148)	0	14	27.8 (809)	0	7	29.1 (927)	0	29
Remote Homo	36.1 (233)	0	1	23.4 (658)	0	5	31.9 (965)	0	8

BAliBASE (Thompson J. *et al.*, 1999)

	17-20	glo	loc	20-22	glo	loc	22-25	glo	loc
PHYBAL hot-spot	**52.8 (2189)**	161	41	57.9 (1054)	0	80	75.0 (3737)	120	42
PHYBAL-2D + IHBM & OHBM	49.4 (1971)	11	13	57.0 (1048)	0	16	72.1 (3614)	10	16
Gonnet	51.9 (2025)	10	6	55.5 (1094)	0	2	73.3 (3750)	19	16
HSDM	48.6 (1934)	1	5	54.8 (986)	0	10	**75.9 (3813)**	21	3
Blosum 62	49.8 (1993)	7	7	**63.4 (1279)**	2	1	74.9 (3852)	18	10
Blosum 45	49.1 (1917)	8	10	57.1 (1087)	0	6	72.6 (3702)	8	10
Blosum 30	45.3 (1884)	0	17	54.5 (856)	0	2	71.7 (3676)	7	11
PAM250	42.4 (1629)	0	3	54.0 (1101)	0	2	71.0 (3604)	1	8
Johnson	46.4 (1916)	0	1	55.8 (1092)	0	2	72.0 (3670)	6	8
Remote Homo	38.5 (1626)	0	7	48.7 (1042)	0	4	66.2 (3372)	0	14

Matrices validation : 2 matrices **vs** 1 matrix in 4-dim gap space

Method		≤ 12	12-15	15-17	17-20	20-22	22-25
PHYBAL	PCAP	**45.0**	**39.3**	**40.1**	**53.3**	59.9	75.1
	CAP	277	1100	1289	2164	1084	3755
	glo	570	477	1214	1003	0	282
	loc	32	102	327	185	107	132
PHYBAL hot-spot	PCAP	43.2	39.0	**40.1**	52.8	57.9	75.0
	CAP	296	1203	1289	2189	1054	3737
	glo	10	11	150	161	0	61
	loc	3	3	21	4	80	42
PHYBAL+Blosum62&Blosum62	PCAP	33.8	33.2	37.7	52.5	**65.7**	**77.6**
	CAP	228	1032	1258	2095	1269	3901
	glo	0	0	98	1030	58	1808
	loc	67	273	26	40	3	114

Two main results come out of this analysis:

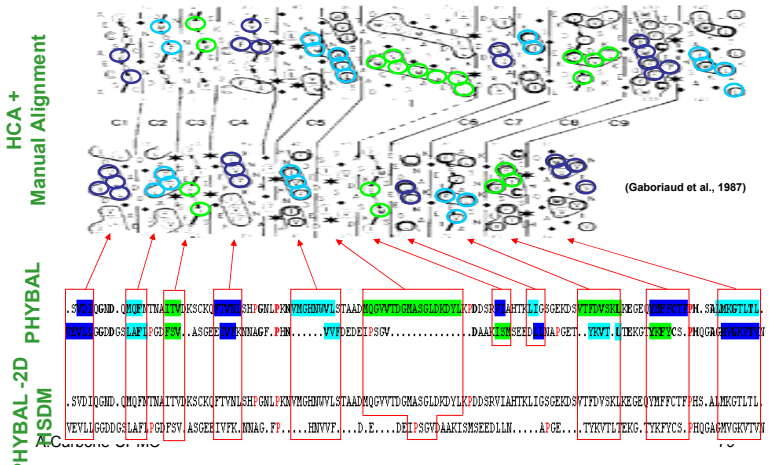
2 matrices describing hydrophobic blocks evolution

4 dimensionality of the gap space

improve pairwise alignment of distantly related proteins

Comparison with Hydrophobic Cluster Analysis method

Plastocyanin – Azurin (β , 13% Id)



Comparison with HMMSUM

Method		≤ 12	12-15	15-17	17-20	20-22	22-25
PHYBAL hot-spot	PCAP	**44.9**	**39.4**	**40.3**	52.8	57.9	75.0
	CAP	**257**	**1037**	1231	**2189**	1054	3737
	glo	17	18	78	93	167	74
	loc	2	2	22	41	80	42
HMMSUM	PCAP	38.8	35.1	38.0	**54.3**	**59.2**	**76.9**
	CAP	200	927	**1278**	2062	**1138**	**3836**
	glo	0	4	5	43	24	43
	loc	12	8	5	16	14	20

HMMSUM predicts secondary structures and aligns by using 281structural context-based aa substitution matrices

(Huang and Bistroff, 2006)

Back to the motivations

- Detection of homologous proteins
- Detection of protein binding sites
- Multiple alignment

Weak Homology : some difficult cases

The length-dependency problem

```
1fleI    .AQEPVKGPVSTKPGSCPTILIRCAMLNPPNRCLKD.TDCPGIKKCCEGSCGMACFVPQ....
1a0aA    MKRESHKHAEQARRNRLAVFLSEIASLIPAEWKQQNVSAAPSKATTVEAACRYIRHLQQNGST
```

Small blocks of identities

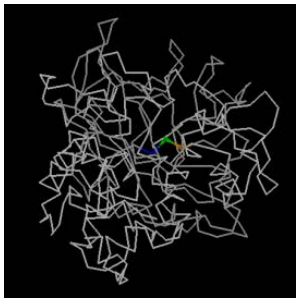
```
1fleI     AQEPVKGPVSTKPGSCPTILIRCAMLNPPN....RCLKDTDCPGIKKCCEGSCG.MACFVP.Q
1udkA     .........NEKSGSCPDMSM...PIPPLGICKTLCNSDSGCPNVQKCCKNGCGFMTCTTPVP
```

Non homologous
with some small blocks of identities

```
1cnsA  SVSSLLSRAQFDRMLLHRNDGACQAKGFYTYDAFVAAAAASGSGTTGSADVQKREVAAFLAQTSHE.TTGG.WATAPDGAFAWGYCFKQERGASSDYCT
1j83A  ...QPTAPKDFSGFWDPNDGTTQ..GFGV.....NPDSPTTAINVENANN..ALKGSNLNSKGSNDLSEGNTWANVRISADIWG....QSINIYGD..T

       PSAQWPCAPGKFCGRGPIQSSHNYNYGPAGRAIGVDLLANPDLVATDATVSFKTAIWFWMTAQPPKPSSHAFIVGQWSPSGADRAAGRVPGFGVITNII
       KLTMDVIAP.......TPVNSS..LSAIPQSSTHG...WGNP.......TRAIR..VWTNNFVAQTDGTYKATLTISTNDSPNFNPIATDAADSVVTNMI

       NSGCECGHGQDSRVADRIGFYKRYCDILGVGYGNNLDCYSQRPFA
       ...LFVGSNSDNISLDNIKFTK.......................
```



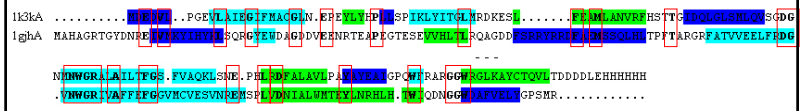
Similar score to weakly homologous proteins

Homologous
with no block of identities

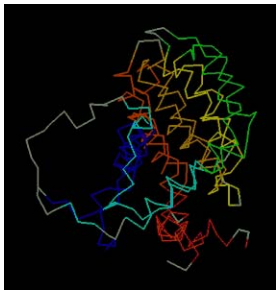
```
1bwwA  TPDLQTQSPSSLSASVGDRVTITCQASQ..DIIKYLLWYQQKPGKAPKLLIYEASNLQAGVPSRFSGSGSGTDYTFTISSLQPEDIATYYCQQMDN.P
1cdcA  .......RDSGTWGGALGHGNNNTPNFQNTDDIDEVRW..ERGSTLVAEFKRKHKPDIK..SGAFEILANG...DLKIKMLTRDDSGTKVTVVSTNGT

       YTFGQGTKLQIT.
       RILDKALDLRILE
```

Non-homologous : but partial homology



Globin-like domain in common



Références bibliographiques

D.S.Hirschberg, Algorithms for the longest common subsequence problem, *J. ACM*, 24:664-675, 1977.

D.Gusfield, *Algorithms on strings, Trees and Sequences*, Cambridge University Press, 1997.

J.Baussand,C.Deremble, A.Carbone, Periodic distributions of hydrophobic amino acids allows the definition of fundamental building blocks to align distantly related proteins, *Proteins: Structures, Functions and Bioinformatics*, 2006.

Autres références

P.Clote et R.Backofen, *Computational Molecular Biology*, Wiley Series, Mathematical and Computational Biology, Princeton University Press, 2001.

R.Durbin, S.Eddy, A.Krogh, G.Mitchison, *Biological Sequence Analysis, probabilistic models of proteins and nucleic acids*, Cambridge University Press, 2000.

P.Pevzner, *Computational Molecular Biology, an algorithmic approach*, MIT Press, 2000.

D.Graur et Wen-Hsiung Li, *Fundamentals of Molecular Evolution*, Sinauer, second edition, 1999.

I.Kirching, P.Forey, Ch. Humphries, D.Williams, *Cladistics, the theory and practice of parsimony analysis*, Oxford University Press, second edition, 2000.

C.Branden et J.Tooze, *Introduction to Protein Structure*, Garland Publishing, second edition, 1999.