Table des matières

Alignement global de deux séquences

Présentation du problème de l'alignement global

Deux séquences S1 et S2, c'est-à-dire deux suites de caractères choisis dans un alphabet A, sont données. Elles sont *a priori* de longueurs différentes L1 et L2. L'alphabet est quelconque ; par exemple, les biologistes utilisent l'alphabet à quatre lettres {A, T, G, C} représentant les quatre bases de l'ADN ou un alphabet à vingt lettres représentant les acides aminés des protéines.

Effectuer un alignement global de ces deux séquences, c'est écrire en regard l'une de l'autre deux lignes de même longueur contenant ces deux suites de caractères de façon qu'elles soient le plus possible similaires. Si les deux séquences sont identiques, un alignement parfait est simplement obtenu en écrivant les deux séquences l'une en dessous de l'autre. Si les deux séquences ne sont pas identiques, un alignement ne pourra être construit qu'en tolérant des différences qui peuvent être des « trous » introduits dans l'une ou l'autre séquence ou bien une différence de caractères.

Par exemple, pour les deux séquences suivantes de longueurs L1=7 et L2=6,

```
S1:  ATACTGA
S2:  TAGATA
```

l'alignement suivant de longueur 8 :

```
N° de colonne  :    12345678
Séquence 1     :    ATAC-TGA
Séquence 2     :    -TAGAT-A
```

réalise une correspondance de 4 caractères (soit 50% des positions de l'alignement), au prix d'un trou « - » dans la séquence 1, de 2 trous dans la séquence 2, et d'une différence tolérée entre le C et le G de la colonne 4. Plusieurs alignements sont réalisables pour chaque paire de séquences; ainsi, le suivant:

```
N° de colonne  :    12345678
Séquence 1     :    -ATACTGA
Séquence 2     :    TAGA-T-A
```

de même longueur, réalise également 50% de correspondances, avec le même nombre de trous et de différences de caractères (ici une en colonne 3 associant un G au T), tandis que le suivant:

```
N° de colonne  :    123456789
Séquence 1     :    -ATAC-TGA
Séquence 2     :    TAGAT---A
```

occupe 9 positions pour réaliser seulement 3 concordances au prix de 5 trous et des différences de caractères dans deux colonnes.

En biologie, en prenant comme référence la séquence 1, un trou introduit dans la ligne du haut sert à y ménager un espace pour insérer en regard un caractère de la séquence 2; cette opération s'appelle donc une *insertion*. Un trou introduit dans la ligne de la séquence 2 signifie qu'il faut « supprimer » un caractère de la séquence 1 pour ne pas avoir de différence avec la 2; cette opération reçoit le nom de *délétion*. Une différence de caractères conservée dans l'alignement s'appelle une *substitution* (et peut s'expliquer par une mutation ou par une occurrence aléatoire). Une concordance est en général désignée par le vocable *identité.*

Un alignement sera considéré comme bon s'il fait concorder un nombre élevé de positions avec une longueur faible et un nombre minimal d'insertions, de délétions et de substitutions. Ceci conduit naturellement à l'idée d'évaluer la qualité d'un alignement en lui attribuant une note construite en donnant une prime à l'alignement pour chaque identité et une pénalité pour chaque opération de modification. La notation de l'alignement considéré peut ainsi être calculée en sommant les primes d'identité C et les pénalités I d'insertion, D de délétion et M de substitution effectuées. I, D et M sont des nombres judicieusement choisis par l'évaluateur. Cette note représente la *similarité* des deux séquences[1].

1 pour qu'elle représente au contraire la différence entre elles, il est possible de prendre, par exemple, C nul et I, D, M positifs

Dès lors, rechercher le(s) meilleur(s) alignement(s) revient à rechercher le(s) alignement(s) réalisant le meilleur score, c'est-à-dire recevant la note *la plus haute*, indiquant la plus grande similarité possible. Le résultat de l'alignement dépend bien sûr des valeurs C, I, D et M[2].

Dans l'exemple ci-dessus, si la prime d'identité est C=0 et si les pénalités élémentaires sont choisies comme suit: I=D=-1 et M=-2, les deux premières solutions ont une similarité de 3*(-1)+1*(-2)=-5 tandis que la seconde obtient un score de 5*(-1)+2*(-2)=-9, traduisant la moins bonne qualité de l'alignement.

Pour des séquences très longues comme celles rencontrées en biologie, le temps de calcul de tous les alignements possibles est rédhibitoire, d'où la nécessité d'algorithmes optimisés.

Programmation dynamique de l'alignement global de deux séquences

Comme l'exemple traité plus haut l'indique, il serait possible d'élaborer une méthode pour construire tous les alignements réalisables, puis de calculer le score de chacun. Ceci demanderait un volume de mémoire et un temps de calcul importants. Pour éviter cela, une méthode élégante a été mise au point. Les premiers à l'avoir utilisée sont Needleman et Wunsch [NW 1]. Elle fait appel à la programmation dynamique dans la mesure où elle effectue l'évaluation de tous les alignements pas à pas, en remplissant une matrice de similarité de manière progressive, en progressant par récurrence.

Principe de base

L'idée de base est d'aligner successivement des sous-séquences de longueurs croissant de 0 à la longueur du résultat final, situées au début des deux séquences à aligner, en maximisant le score du sous-alignement à chaque étape (c'est là que se trouve réellement le caractère *dynamique* de la résolution, le critère étant maximisé *terme par terme*).

Le procédé utilise une matrice qui permet d'enregistrer les scores de tous les sous-alignements. Chaque case de la matrice doit contenir le score le plus élevé réalisable pour le sous-alignement correspondant. La dimension horizontale de la matrice est associée à la première séquence et la dimension verticale à la seconde. Comme la première longueur de sous-séquences considérée est nulle, il faut prévoir une ligne et une colonne d'indices nuls. Pour numéroter les lignes et les colonnes, les indices utilisés sont donc i= 0, ..., L1 et j= 0, ..., L2 (attention, comme ici, dans la suite de ce texte, le premier indice cité est celui de la colonne et le second celui de la ligne en cours).

Dans le cas de l'exemple ci-dessus, la matrice se présente comme suit.

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

Figure 1: la matrice vide

Comme plus haut, on prend une prime d'identité nulle (C=0) et les pénalités I=D= -1 et M= -2.

2 mais changer les quatre signes pour rendre minimale la différence au lieu de maximiser la similarité conduit au même résultat.

Initialisation de la matrice

La case (0,0) doit contenir le meilleur score possible pour l'alignement de deux séquences de longueurs nulles. Deux telles séquences sont nécessairement identiques donc il n'y a besoin d'aucune opération pour les aligner et, en conséquence, la case (0,0) de la matrice contient nécessairement un score nul.

La case (1,0) contient le meilleur score possible pour l'alignement de la sous-séquence de longueur 1 de S1, soit A, avec la sous-séquence de longueur nulle de S2. Il n'y a qu'une manière de faire cela, par délétion du caractère A et cela donne l'alignement suivant, de longueur 1:

```
N° de colonne  :    1
Séquence 1     :    A
Séquence 2     :    -
```

avec la pénalité d'une délétion, soit **-1** avec les valeurs prises pour cet exemple.

La case (2,0) contient le meilleur score possible pour l'alignement de la sous-séquence de longueur 2 de S1, soit AT, avec la sous-séquence de longueur nulle de S2. Il n'y a qu'une manière de faire cela, par délétion des deux caractères A et T et cela donne l'alignement suivant, de longueur 2:

```
N° de colonne  :    12
Séquence 1     :    AT
Séquence 2     :    --
```

avec la pénalité de deux délétions, soit **-2** avec les valeurs prises pour cet exemple.

Et ainsi de suite jusqu'à la fin de la ligne j=0 qui contient donc les valeurs 0, -1, -2, -3, -4, -5, -6, -7.

De manière similaire, la case (0,1) contient le meilleur score possible pour l'alignement de la sous-séquence de longueur nulle de S1 avec la sous-séquence de longueur 1 de S2, soit T. Il n'y a qu'une manière de faire cela, par insertion du caractère T et cela donne l'alignement suivant, de longueur 1:

```
N° de colonne  :    1
Séquence 1     :    -
Séquence 2     :    T
```

avec la pénalité d'une insertion, soit **-1** avec les valeurs prises pour cet exemple.

La case (0,2) contient le meilleur score possible pour l'alignement de la sous-séquence de longueur nulle de S1 avec la sous-séquence de longueur 2 de S2, soit TA. Il n'y a qu'une manière de faire cela, par insertion des deux caractères T et A et cela donne l'alignement suivant, de longueur 2:

```
N° de colonne  :    12
Séquence 1     :    --
Séquence 2     :    TA
```

avec la pénalité de deux insertions, soit **-2** avec les valeurs prises pour cet exemple.

Et ainsi de suite jusqu'à la fin de la colonne i=0 qui contient donc les valeurs 0, -1, -2, -3, -4, -5, -6.

A ce stade, il n'y a pas eu encore d'optimisation puisqu'il n'y avait qu'une solution possible pour chaque case. L'étape effectuée est donc seulement *l'initialisation* de l'algorithme. Cette initialisation donne à la matrice l'allure suivante.

	i=	0	1	2	3	4	5	6	7
j=			A	T	A	C	T	G	A
0		0	-1	-2	-3	-4	-5	-6	-7
1	T	-1							
2	A	-2							
3	G	-3							
4	A	-4							
5	T	-5							
6	A	-6							

Figure 2: la matrice initialisée

Calcul dynamique de l'alignement

La matrice va être remplie ligne après ligne, en progressant de gauche à droite sur chaque ligne, comme pour la lecture d'un texte.

Ainsi, la case (1,1) doit contenir le meilleur score possible pour l'alignement de la sous-séquence de longueur 1 de S1, soit A, avec la sous-séquence de longueur 1 de S2, soit T. Il y a en fait trois manières de réaliser cela en s'appuyant sur les alignements déjà évalués:

- la première consiste à partir du sous-alignement de longueur nulle des deux sous-séquences de longueurs nulles correspondant à la case (0,0) et d'inscrire à sa suite les deux premiers caractères des deux séquences, ce qui donne l'alignement suivant de longueur 1:

```
N° de colonne  :    1
Séquence 1     :    A
Séquence 2     :    T
```

Comme les deux caractères sont différents, l'acceptation de cet alignement supposerait celle d'une substitution M=-2 dont la pénalité s'ajouterait à celles des opérations déjà évaluées dans la case (0,0), soit 0. **Le score de cette solution est donc 0+(-2) = -2** (si les deux caractères s'étaient avérés identiques, il n'y aurait pas eu de substitution à effectuer pour assurer l'alignement et le score à retenir pour cette solution aurait été 0+0=0).

- La deuxième solution consiste à partir du sous-alignement de longueur 1 correspondant à la case (1,0) située au-dessus et à prendre en compte le caractère suivant de la séquence 2 (c'est-à-dire le premier), soit T, ce qui permet de passer à un nouvel alignement de longueur 2:

```
N° de colonne  :    1          12
Séquence 1     :    A    =>    A-
Séquence 2     :    -          -T
```

Cette solution suppose donc une insertion de pénalité -1 qui s'ajoute au score déjà établi pour la case (1,0), soit -1, **d'où un score total de cette option de (-1)+(-1) = -2**. A la différence de l'option précédente, même si les deux caractères sont semblables, il n'y a pas de réduction de pénalité à espérer!

- Enfin la troisième solution consiste à partir du sous-alignement de longueur 1 correspondant à la case (0,1) située à gauche et à prendre en compte le caractère suivant de la séquence 1 (son premier en fait), soit A, ce qui permet de passer à un nouvel alignement de longueur 2:

```
N° de colonne  :    1          12
Séquence 1     :    -    =>    -A
Séquence 2     :    T          T-
```

qui suppose une délétion de pénalité -1 qui s'ajoute au score déjà établi pour la case (0,1), soit -1, **d'où un score total de cette option de (-1)+(-1) = -2**.

En principe, l'option retenue est celle qui donne le résultat de score incrémenté le plus grand. Ici les trois solutions sont équivalentes en score. Le nombre à écrire en case (1,1) est donc -2 et il est possible de

retenir que les trois variantes sont admissibles car de même score.

Ensuite, le remplissage de la matrice se fait en progressant de proche en proche, toujours de la même façon. Pour trouver le coût C_{ij} à inscrire en case (i,j), les trois manières sont comparées:

- partir de la case au-dessus à gauche (i-1,j-1) et ajouter à l'alignement correspondant sur la ligne du haut le $i^{ème}$ caractère de la séquence S1, soit S1[i], et sur la ligne du bas le $j^{ème}$ caractère de la séquence S2, soit S2[j]. Si S1[i]=S2[j] (identité), le nouvel alignement est réalisé sans changer le score : $C'_{ij} = C_{(i-1)(j-1)}$, tandis que, si S1[i]<>S2[j], l'alignement suppose une substitution: $C'_{ij} = C_{(i-1)(j-1)}+M$;
- partir de la case au-dessus et effectuer une insertion en plaçant sur la ligne du bas le $j^{ème}$ caractère de la séquence S2, soit S2[j], en face d'un trou; le score candidat correspondant est $C''_{ij} = C_{i(j-1)} + I$;
- partir de la case à gauche et introduire une délétion en plaçant sur la ligne du haut le $i^{ème}$ caractère de la séquence S1, soit S1[i], en regard d'un trou; le score associé est $C'''_{ij} = C_{(i-1)j} + D$;
- choisir pour C_{ij} la plus petite des trois valeurs:

$$C_{ij} = \min(C'_{ij}, C''_{ij}, C'''_{ij})$$

et retenir la (ou les) opération(s) correspondante(s).

Ce pas de traitement peut être représenté graphiquement de la manière suivante.

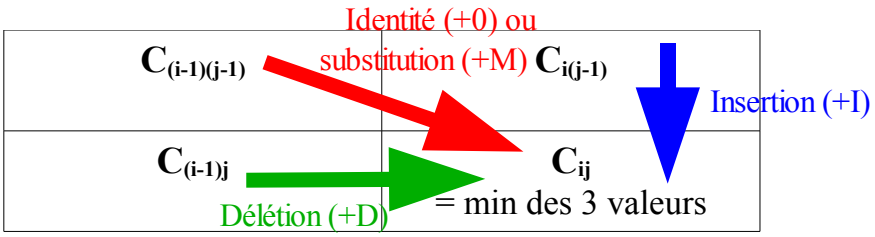

Figure 3: détermination du score en case (i,j)

A la fin du traitement, lorsque l'alignement de toute la séquence est effectué, la case (L1, L2) de la matrice contient le score total de l'alignement et la « remontée » des opérations effectuées fournit la succession des transformations effectuées (et de leur variantes éventuelles) et donc l'alignement résultant.

	i=	0	1	2	3	4	5	6	7
j=			A	T	A	C	T	G	A
0		0	-1	-2	-3	-4	-5	-6	-7
1	T	-1	-2 (3 sol)	-2 (al)	-3 (dél)	-4 (dél)	-4 (al)	-5 (dél)	-6 (dél)
2	A	-2	-1 (al)	-2 (dél)	-2 (al)	-3 (dél)	-4 (dél)	-5 (dél)	-5 (al)
3	G	-3	-2 (ins)	-3 (3 sol)	-3 (ins)	-4 (3 sol)	-5 (3 sol)	-4 (al)	-5 (dél)
4	A	-4	-3 (al)	-4 (3 sol)	-3 (al)	-4 (dél)	-5 (dél)	-5 (ins)	-4 (al)
5	T	-5	-4 (ins)	-3 (al)	-4 (ins, dél)	-5 (3 sol)	-4 (al)	-5 (dél)	-5 (ins)
6	A	-6	-5 (al)	-4 (ins)	-3 (al)	-4 (dél)	-5 (ins, dél)	-6 (3 sol)	-5 (al)

Figure 4: la matrice finale

Appliqué à l'exemple donné plus haut, l'algorithme produit la matrice finale de la figure 4. Dans chaque case, sous le score obtenu, le commentaire entre parenthèses indique le chemin qui permet

d'obtenir ce résultat avec la signification suivante :

(al) : pas fait depuis la case en haut à gauche (diagonale), sans substitution (score inchangé)
(ins) : pas fait depuis la case au-dessus (score + I)
(dél) : pas depuis la case à gauche (score + D)
(sub) : pas fait depuis la case en haut à gauche (diagonale), avec substitution (score + M)
(ins, dél) : (ins) et (dél) sont équivalents
(3 sol) : (ins), (dél), (sub) sont équivalents.

Grâce à ces indications, en partant de la case (L1, L2) qui contient le score maximisé (-5), le chemin parcouru se trace en plaçant les flèches correspondantes en remontant vers l'origine.

	i=	0	1	2	3	4	5	6	7
j=			A	T	A	C	T	G	A
0		0	-1	-2	-3	-4	-5	-6	-7
1	T	-1 (ins)	-2 (3 sol)	-2 (al)	-3 (dél)	-4 (dél)	-4 (al)	-5 (dél)	-6 (dél)
2	A	-2	-1 (al)	-2 (dél)	-2 (al)	-3 (dél)	-4 (dél)	-5 (dél)	-5 (al)
3	G	-3	-2 (ins)	-3 (3 sol)	-3 (ins)	-4 (3 sol)	-5 (3 sol)	-4 (al)	-5 (dél)
4	A	-4	-3 (al)	-4 (3 sol)	-3 (al)	-4 (dél)	-5 (dél)	-5 (ins)	-4 (al)
5	T	-5	-4 (ins)	-3 (al)	-4 (ins, dél)	-5 (3 sol)	-4 (al)	-5 (dél)	-5 (ins)
6	A	-6	-5 (al)	-4 (ins)	-3 (al)	-4 (dél)	-5 (ins, dél)	-6 (3 sol)	-5 (al)

Figure 5: la matrice finale complétée par le chemin suivi

Noter que pour passer de la case (1, 2) à la case (2, 3), trois chemins sont équivalents, ce qui signifie que trois alignements sont équivalents en ce sens qu'ils ont le même score. Pour les construire, il suffit de suivre les flèches depuis l'origine. Partant de l'alignement vide, le chemin est constitué successivement de:

- l'insertion du caractère T de S2,
- l'identité des caractères A de S1 et S2,
- l'un des trois mouvements suivants équivalents:
  - placement du T de S1 avec délétion suivi de l'insertion du G de S2,
  - introduction de substitution : T de S1 en face de G de S2,
  - insertion du G de S2 suivi du placement du T de S1 avec délétion,
- identité des deux A de S1 et S2,
- placement du C de S1 avec délétion,
- identité des T de S1 et S2,
- placement du G de S1 avec délétion,
- et, enfin, identité des A de S1 et S2.

Les trois alignements équivalents s'écrivent donc (en gras, les éléments différents selon les variantes) :
Variante 1, alignement de longueur 9, 4 identités (44%)

```
N° de colonne  :    123456789
Séquence 1     :    -A**T**-ACTGA
Séquence 2     :    TA-**G**A-T-A
```

Variante 2, alignement de longueur 8, 4 identités (50%)

```
N° de colonne  :    12345678
Séquence 1     :    -ATACTGA
Séquence 2     :    TAGA-T-A
```

Variante 3, alignement de longueur 9, 4 identités (44%)

```
N° de colonne  :    123456789
Séquence 1     :    -A-TACTGA
Séquence 2     :    TAG-A-T-A
```

Commentaires

- Si la variation de score n'est pas nulle, parmi des pas équivalents, celui en diagonale (substitution) est le plus court et conduit au pourcentage d'identités le plus élevé (la longueur du chemin est au dénominateur de la fonction exprimant le pourcentage). C'est le cas de la variante 2 dans l'exemple ci-dessus.

Alignement local de deux séquences

Présentation du problème de l'alignement local

La réalisation d'un alignement global de deux séquences permet de trouver les ressemblances entre deux séquences assez similaires. Cependant dans certains cas, le but visé est plutôt de trouver, dans deux séquences assez différentes, des régions similaires éventuellement séparées par des régions de faible similitude, opération désignée comme un alignement local, par exemple pour trouver des motifs communs entre des séquences relativement distinctes.

Dans ce cas, la méthode d'alignement global décrite plus haut ne donne en général pas de bons résultats puisqu'elle étire les deux séquences. L'algorithme de programmation dynamique de Smith et Waterman [SW 1] a été développé pour répondre au besoin d'alignement local. C'est une adaptation de la méthode de Needleman et Wunsch, modifiée pour favoriser l'identification de régions communes.

Programmation dynamique de l'alignement local de deux séquences

Principe

L'algorithme de Smith et Waterman peut être vu comme une variante de celui de Needleman et Wunsch. Pour la recherche de l'alignement local, ce sont les identités qui doivent être privilégiées. En revanche, on doit admettre l'insertion de large trous. L'approche adoptée privilégie donc la similarité.

Pour privilégier les identités par rapport aux insertions-délétions, il est d'abord utile de choisir une valeur non nulle pour la « prime » de concordance C de deux caractères (identité). Chaque cellule de la matrice contient donc un score qui peut être positif ou négatif suivant l'opération faite pour y arriver.

Ensuite, après le calcul dynamique des scores, toutes les cases de la matrice qui contiennent un score négatif sont mises à zéro. La case de la matrice qui contient la valeur la plus élevée correspond à l'alignement optimal et la construction de cet alignement se fait en partant de cette case et en remontant jusqu'à la première case contenant un zéro.

Exemple

Un alignement local des séquences TGAGATCATG et AGAT est recherché. En prenant comme valeur de prime C = 3 pour l'identité et comme valeurs de pénalités: -1 pour la substitution et -2 pour l'insertion et la délétion, la matrice de similarité se construit par étapes, comme pour l'alignement global:

- initialisation de la ligne j=0 avec des délétions (scores de -2 enchaînés) et de la colonne i=0 avec des insertions (-2 également),
- puis, pour chaque cellule, comparaison des scores obtenus par insertion (descente verticale de la case au dessus), délétion (pas vers la droite depuis la cellule à gauche) ou substitution/identité (pas descendant vers la droite en diagonale, avec un score élémentaire dépendant de l'identité ou non des deux caractères ajoutés) et choix de l'opération rendant maximal le score de la cellule.

La matrice de similarité se remplit comme le montre la figure 6.

	Colonne i =	0	1	2	3	4	5	6	7	8	9	10
Ligne j=	Séquence 1		**T**	**G**	**A**	**G**	**A**	**T**	**C**	**A**	**T**	**G**
0	Séquence 2	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20
1	**A**	-2	-1	-3	-1	-3	-5	-7	-9	-11	-13	-15
2	**G**	-4	-3	2	0	2	0	-2	-4	-6	-8	-10
3	**A**	-6	-5	0	5	3	5	3	1	-1	-3	-5
4	**T**	-8	-3	-2	3	4	3	8	6	4	2	0

Figure 6: matrice de similarité

La matrice de la figure 7 recueille les opérations qui ont permis d'obtenir ces scores (la présence de deux lettres indique l'équivalence en score de deux opérations).

	Colonne i =	0	1	2	3	4	5	6	7	8	9	10
Ligne j=	Séquence 1		**T**	**G**	**A**	**G**	**A**	**T**	**C**	**A**	**T**	**G**
0	Séquence 2		D	D	D	D	D	D	D	D	D	D
1	**A**	I	M	DM	C	D	DC	D	D	DC	D	D
2	**G**	I	IM	C	D	C	D	D	D	D	D	DC
3	**A**	I	IM	I	C	D	C	D	D	DC	D	D
4	**T**	I	C	I	I	M	I	C	D	D	DC	D

Figure 7: la matrice des opérations retenues

L'étape suivante est d'annuler tous les cases comportant un score négatif, de repérer la cellule de score maximal (8 dans le cas particulier) et remonter de cette cellule jusqu'à une case de contenu nul en se servant de la matrice précédente, comme indiqué par les flèches portées sur la figure 7.

	Colonne i =	0	1	2	3	4	5	6	7	8	9	10
Ligne j=	Séquence 1		**T**	**G**	**A**	**G**	**A**	**T**	**C**	**A**	**T**	**G**
0	Séquence 2	0	0	0	0	0	0	0	0	0	0	0
1	**A**	0	0	0	0	0	0	0	0	0	0	0
2	**G**	0	0	2	0	2	0	0	0	0	0	0
3	**A**	0	0	0	5	3	5	3	1	0	0	0
4	**T**	0	0	0	3	4	3	8	6	4	2	0

Figure 8: mise en évidence du résultat obtenu pour l'alignement local

Les cases sélectionnées visualisent dans la figure 8 l'alignement local obtenu qui est donc le suivant:

```
N° de colonne  :    123456789₁₀
Séquence 1     :    TGAGATCATG
Séquence 2     :      AGAT    .
```

Critique des méthodes d'alignement précédentes

Les deux méthodes d'alignement présentées possèdent l'avantage d'être exactes. Pour les valeurs de primes et pénalités choisies, le critère est exactement calculé et, si l'algorithme est bien programmé, il donne les différentes solutions optimales pour ces valeurs. Elles ne sont cependant pas exemptes d'inconvénients dont les deux principaux sont:

- le caractère sommaire du système de notation adopté: 1) il est le même tout le long de la séquence; 2) la pénalité associée à la création d'une brèche (insertion ou délétion) est proportionnelle à la longueur de celle-ci alors qu'une fois la brèche créée son allongement devrait être moins coûteux; 3) la pénalité associée à une mutation ne dépend pas de la paire de caractères alignée, ce qui est encore plus gênant pour les séquences protéiques que pour les séquences nucléiques;
- ensuite, l'emploi de l'algorithme exact convient à l'alignement de deux séquences mais, malgré son optimisation, son emploi demande un temps rédhibitoire lorsqu'il s'agit de procéder à des comparaisons nombreuses ou à l'alignement d'un ensemble de multiples séquences.

Pour pallier ces inconvénients, des solutions existent. Le premier peut être levé en employant des systèmes de score plus sophistiqués. Le second conduit à abandonner la résolution exacte pour recourir à des méthodes heuristiques approximatives mais fournissant rapidement un résultat de bonne qualité.

Systèmes de scores

Le score des opérations d'alignement de séquences utilisées plus haut (identité, substitution, insertion, délétion) n'est pas très significatif du point de vue biologique.

Matrices nucléiques

Ainsi, les substitutions d'acides nucléiques pénalisent toutes les mutations de la même manière. Cependant, bien que les transitions possibles soient deux fois moins nombreuses que les transversions, elles se produisent plus fréquemment et devraient donc être moins pénalisées que les transversions.

Le remède consiste à noter les opérations en utilisant une matrice de substitution. Dans l'exemple de la figure 6, les identités sont notées 3 et les substitutions -1, quelles qu'elles soient. Une autre manière d'exprimer cela est de dire que la matrice de substitution correspondante est celle de la figure 10, appelée pour cette raison « matrice identité » (où seule la diagonale principale se distingue, un peu comme dans la matrice « unité » des mathématiques).

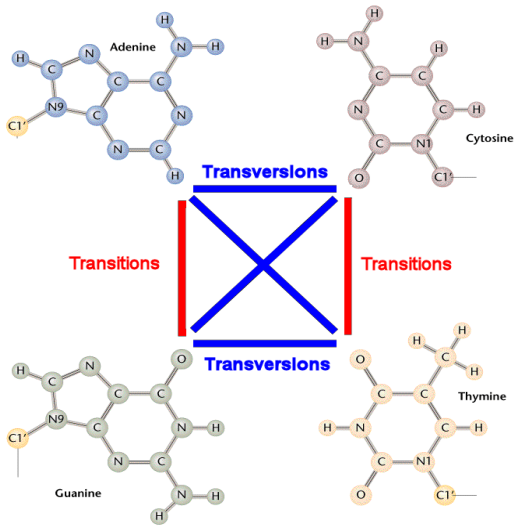

Figure 9: transitions et transversions

	A	C	G	T
A	3	-1	-1	-1
C	-1	3	-1	-1
G	-1	-1	3	-1
T	-1	-1	-1	3

Figure 10: matrice identité

Chaque cellule donne le score à affecter à l'alignement de deux caractères. Il suffit de changer les valeurs pour obtenir une matrice de substitution plus nuancée. Ainsi, la matrice de la figure 11 donne un score de 3 aux identités, de 1 aux transitions et de -1 aux transversions. Le sens de ses deux dernières opérations ne faisant pas de différence, la matrice est symétrique (une substitution de G en A reçoit le même score que la substitution symétrique de A en G).

	A	C	G	T
A	3	-1	1	-1
C	-1	3	-1	1
G	1	-1	3	-1
T	-1	1	-1	3

Figure 11: matrice à trois valeurs identité=3/transition=1/transversion=-1

Matrices protéiques

Un système basé uniquement sur l'identité est encore moins approprié pour les séquences protéiques.

En effet, un acide aminé peut être remplacé par un autre sans que la structure ou la fonctionnalité d'une

protéine soit grandement altérée. Les acides aminés peuvent donc être groupés en familles et ceci permet de bâtir des systèmes de scores tenant compte de la réalité biologique et améliorant la fiabilité des recherches de similitudes protéiques.

Une des premières matrices à utiliser ce principe a été celle déduite de la dégénérescence du code génétique, les scores élémentaires étant fonctions des nucléotides présents dans les codons des acides aminés. De nombreuses autres matrices ont été considérées et constituent deux grandes classes:

- celles fondées sur les caractéristiques physico-chimiques des acides aminés,
- celles exploitant l'évolution moléculaire des protéines.

Les premières sont conçues pour révéler au mieux certaines caractéristiques physico-chimiques communes à deux protéines, comme le caractère hydrophile ou hydrophobe des protéines ou leur structure secondaire ou tertiaire. Elles ne sont pas étudiées ici.

Les secondes sont construites à partir de substitutions observées entre acides aminés au cours de l'évolution moléculaire, déterminées à partir d'alignements multiples de protéines homologues. Les fréquences ainsi mise en évidence sont représentatives des probabilités de substitutions altérant peu les fonctions de la protéine. La connaissance de ces probabilités permet de construire des matrices de substitutions adaptées à la mise en évidence de relations d'homologie. Deux grandes catégories de matrices se sont dégagées: les PAM et les BLOSUM. Les deux sont élaborées selon les mêmes principes probabilistes, comme suit.

Chaque séquence protéique est considérée comme une longue chaîne de caractères aléatoires, indépendants et identiquement distribués (hypothèse simplificatrice). Deux séquences S et T sont considérées. La probabilité uniforme de trouver le caractère a en position i serait:

$$P\{S[i] = a\} = \pi_a \quad \text{(constante, c'est-à-dire indépendante de i).}$$

Dans une approche évolutive, la séquence T est maintenant supposée être homologue de S, avec l'hypothèse que les mutations sont aléatoires et indépendantes. Soit $p_{a\to a'}$ la probabilité conditionnelle que le caractère en position i soit devenu a' dans T quand il est a dans S :

$$P\{T[i]=a' \mid S[i] = a\} = p_{a\to a'} \quad .$$

Pour trouver l'alignement a → a' en position i du fait de l'évolution, il faut d'abord avoir a en i puis que a ait muté en a', la probabilité de cet alignement est donc: $P_{évol}\{(S[i] = a) \text{ et } (T[i]=a')\} = \pi_a * p_{a\to a'}$.

Si les deux séquences n'étaient pas homologues mais étaient deux chaînes tirées aléatoirement, avec la loi de probabilité uniforme donnée plus haut, pour trouver le même alignement, il faudrait simplement que a se trouve en position i et que a' se trouve également en position i « par pur hasard ». La probabilité de cet alignement serait donc $P_{aléa}\{(S[i] = a) \text{ et } (T[i]=a')\} = \pi_a * \pi_{a'}$.

Le rapport entre ces deux probabilités

$$R(a\to a',i) = \frac{P_{évol}\{(S[i] = a) \text{ et } (T[i]=a')\}}{P_{aléa}\{(S[i] = a) \text{ et } (T[i]=a')\}} = \frac{\pi_a * p_{a\to a'}}{\pi_a * \pi_{a'}} = \frac{p_{a\to a'}}{\pi_{a'}}$$

est un indicateur de la signification biologique de l'alignement de a' en face de a en position i. En effet,

- si ce ratio est supérieur à 1, c'est que la probabilité de cause évolutive (au numérateur) est plus grande que la probabilité d'un appariement aléatoire (au dénominateur) et l'occurrence de cet alignement doit donc recevoir une prime (valeur positive) dans l'évaluation de la similarité des séquences;

- au contraire, un ratio inférieur à 1 indique une plus grande probabilité pour l'appariement aléatoire et la présence de cet alignement doit donc être traduite par une pénalité (valeur négative) dans le calcul de la similarité entre les séquences.

Il ne reste plus qu'à trouver le moyen de traduire cela mathématiquement. Il se trouve que la fonction logarithme fait exactement ce qu'il faut puisqu'elle est nulle lorsque son argument est égal à 1, positive s'il est supérieur à 1 et négative dans le cas contraire. On peut donc choisir de prendre la contribution à la similarité comme proportionnelle à un logarithme du ratio:

$$\Delta Sim(a \rightarrow a',i) = \lambda \log\left(\frac{P_{évol}\{(S[i] = a) \text{ et } (T[i]=a')\}}{P_{aléa}\{(S[i] = a) \text{ et } (T[i]=a')\}} \right) = \lambda \log\left(\frac{p_{a \rightarrow a'}}{\pi_{a'}} \right)$$

où la constante positive de proportionnalité λ a pour seul rôle de faire en sorte que la contribution ΔSim à la similarité[3] prenne des valeurs ni trop petites ni trop grandes, de l'ordre de ±quelques unités (elle est donc choisie en tenant compte de la gamme de probabilités rencontrées).

Pour calculer l'expression du membre de droite, il faut calculer la fraction entrée dans le logarithme. Son dénominateur est très simple: pour les bases nucléiques, une position a une chance sur quatre de contenir un caractère donné ($\pi_{a'}=1/4$) et pour les acides aminés, une chance sur vingt ($\pi_{a'}=1/20$). Le numérateur est plus difficile à obtenir puisqu'il faut aller observer expérimentalement les fréquences des mutations qui se produisent réellement pour les traduire en probabilités $p_{a \rightarrow a'}$. Dans les approches couramment employées aujourd'hui, ces probabilités sont considérées comme indépendantes de la position i et l'expression utilisée est donc en définitive:

$$\Delta Sim(a \rightarrow a') = \lambda \log\left(\frac{p_{a \rightarrow a'}}{\pi_{a'}} \right)$$

et la matrice de substitution appliquée au calcul du critère de similarité n'est autre que l'ensemble des valeurs $\Delta Sim(a \rightarrow a')$ pour tous les paires {a, a'}.

Plusieurs approches existent pour déterminer les fréquences expérimentales $p_{a \rightarrow a'}$ en question. Les deux plus connues conduisent aux matrices dites PAM et BLOSUM.

Les matrices PAM

La construction de ces matrices a utilisé une base de données de 1572 différences dans 71 groupes de protéines très voisines ayant au moins 85% d'identités (protéines homologues et fonctionnellement équivalentes) [MOD 1]. Tous ces écarts ont été répertoriés et entrés dans une matrice énumérant tous les changements possibles d'acides aminés. Ceci a permis de disposer du décompte de toutes les mutations ponctuelles (c'est-à-dires qui se sont produites sur un site, indépendamment les unes des autres) et acceptées, c'est-à-dire conservées parce qu'elle ne changeaient pas trop la fonction de la protéine, (« *point accepted mutations* » en anglais, d'où le nom des matrices PAM) et d'en déduire la probabilité de mutation d'un acide aminé vers un autre.

Plus précisément, deux séquences sont dites séparées par un intervalle d'évolution de 1 PAM si, en moyenne, il y a un acide aminé muté sur 100. Comme les différents acides aminés n'ont pas la même propension à muter, cette valeur moyenne n'est pas la probabilité de mutation de chacun. Dès lors, la matrice dite PAM1 donne en ligne i, colonne j, la probabilité qu'une mutation fasse passer de l'acide aminé i à l'acide aminé j pendant un intervalle d'évolution de 1 PAM. La matrice PAM N, qui donne les probabilités de mutation de i à j (en prenant en compte les mutations successives intermédiaires[4]) pour un intervalle évolutif correspondant à N mutations ponctuelles acceptées en moyenne sur 100 acides aminés, est simplement la matrice PAM 1 élevée à la puissance N (en raison des hypothèses mathématiques faites

3 Nota: la planche 84 du fichier « Alignement de deux séquences.pdf » du cours Pasteur 2008 écrit un peu brièvement que $\Delta Sim(a \rightarrow a')$, notée S_{ij}, « exprime le ratio entre la probabilité que 2 résidus i et j soient alignés par descendance et la probabilité qu'ils soient alignés par chance »; en fait, mathématiquement, cette quantité exprime le logarithme de ce ratio et il faut surtout retenir qu'elle exprime la variation de la similarité biologiquement significative des deux séquences due à l'appariement des deux caractères en jeu.

4 Remarque: en raison de ces mutations multiples successives sur un même site, qui peuvent « défaire » des changements antérieurs, N fois la distance évolutive correspondant à 1 PAM, soit N PAM, ne donne pas N% de mutations acceptées.

sur le processus aléatoire à l'œuvre):

$$\text{PAM N} = (\text{PAM 1})^N .$$

De ces matrices de probabilités, on passe aux matrices de substitution comme indiqué plus haut, d'abord en calculant les ratios à la probabilité uniforme (cela donne la matrice appelée en anglais « *odds matrix* », c'est-à-dire « matrice des chances » - nom peu judicieux en raison de la confusion possible entre « chances » et « probabilités » en français) puis en passant au logarithme (ce qui donne la matrice de substitution proprement dite, appelée en anglais « *log odds matrix* », c'est-à-dire « matrice des logarithmes des chances »).

Par abus de langage, la matrice de substitution est aussi appelée parfois « PAM N » (se servir du contexte pour lever le doute!); on trouve également parfois le nom de « matrice de mutation » (MDM N).

L'expérience a montré que PAM 250 était un bon compromis pour les besoins de l'évaluation de similarité. La matrice de substitution correspondante est montrée par la figure 12.

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

Figure 12: la matrice de substitution correspondant à PAM 250

Les PAM représentent un progrès net par rapport à la matrice identité mais non sans inconvénients:

- les probabilités de mutation sont considérées comme indépendantes de la position du site;
- le jeu de données ayant servi à établir les probabilités était assez petit, ce qui fait que certaines substitutions ont échappé à l'observation et, par ailleurs,
- il ciblait certaines familles de protéines (petites protéines globulaires) ; des travaux ont corrigé ce biais en élargissant la base de données (1621 familles, 16130 séquences, cf. [JTT1]);
- les matrices de rang élevé, associées à de grands intervalles d'évolution, se calculent à partir de celles valables pour les courtes distances par des opérations qui amplifient les erreurs;
- enfin, la méthode reposait sur un alignement global, ce qui a introduit des régions sources de bruit dans les données.

Les matrices BLOSUM

Pour pallier ces inconvénients, en 1992 (14 ans après les PAM), de nouvelles matrices ont été mises au point en exploitant des alignements locaux sans brèches [HH1] tirés de la banque « BLOCKS » (http://blocks.fhcrc.org/) et donc nommées en anglais « *blocks substitution matrix* », soit BLOSUM. Un bloc est le résultat d'un alignement multiple sans brèches de courtes régions protéiques très conservées. L'avantage est que les protéines considérées sont plus éloignées et donc les matrices peuvent être mieux adaptées à la comparaison de protéines plus divergentes.

Les blocs utilisés sont classés en groupes G_n de séquences ayant au moins un certain pourcentage n% d'identités. Ces séquences sont utilisées pour calculer les probabilités d'occurrence de chaque paire d'acides aminés puis, de là, les logarithmes de fréquences relatives et construire ainsi les matrices de substitution comme indiqué plus haut. A chaque pourcentage n% (choisi comme seuil caractérisant un

groupe de blocs) correspond donc une matrice BLOSUM n.

Le grand nombre de séquences utilisées rend les matrices BLOSUM plus précises que les PAM. En effet, la paire remplacée la moins fréquente a été observée 2369 fois sur un total de 1,25 x 10^6 paires exploitées.

Noter que l'indice utilisé pour indexer les matrices est défini en sens inverses pour les PAM et les BLOSUM: la PAM N de grand N est associée à des grandes distances évolutives (protéines très divergentes) tandis que la BLOSUM N de grand N est associée à des séquences ayant un grand pourcentage N d'identités (protéines peu divergentes).

La matrice de substitution BLOSUM 62, qui est souvent utilisée comme valeur moyenne, est montrée par la figure 13.

Ala	A	4																			
Arg	R	-1	5																		
Asn	N	-2	0	6																	
Asp	D	-2	-2	1	6																
Cys	C	0	-3	-3	-3	9															
Gln	Q	-1	1	0	0	-3	5														
Glu	E	-1	0	0	2	-4	2	5													
Gly	G	0	-2	0	-1	-3	-2	-2	6												
His	H	-2	0	1	-1	-3	0	0	-2	8											
Ile	I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V

Figure 13: la matrice de substitution BLOSUM 62 (matrice symétrique, seule une moitié est montrée)

Choix des matrices

Le choix de « la bonne » matrice à employer est très lié au type d'analyse visée. Les matrices BLOSUM sont aujourd'hui plus utilisées que les PAM mais, de manière générique, compte tenu de ce qui précède et de l'expérience acquise, les règles suivantes peuvent être appliquées.

- Pour des séquences assez similaires, il est souhaitable d'utiliser des matrices BLOSUM N grand ou PAM N faible.
- Au contraire, pour des des séquences longues et divergentes, des BLOSUM N faible ou PAM N grand seront préférées.
- Quand on n'a pas d'idée *a priori* sur la divergence, comme par exemple pour confronter une séquence à celles d'une banque de données, il est possible de commencer par une matrice moyenne comme la BLOSUM 62 ou la PAM 120.

Surtout, il n'y a pas UNE matrice idéale et il ne faut pas hésiter à faire des tests avec différentes matrices, les variations des sorties qui apparaissent pouvant être instructives et aider à critiquer les résultats obtenus.

Et lorsqu'il y a de nombreuses comparaisons à faire ?

Les algorithmes présentés ci-avant sont inexploitables dès que le nombre de séquences à comparer augmente quelque peu, en raison de temps de calcul qui deviennent très vite rédhibitoires, même avec des ordinateurs puissants. Cette situation peut se rencontrer dans diverses circonstances, par exemple lorsque l'on cherche, dans une banque de séquences connues, les séquences qui ressemblent à la séquence qu'on a en main (séquence « requête »), ou bien lorsque plusieurs séquences données doivent être alignées simultanément (alignement multiple).

Pour répondre à ces types de besoins, il faut utiliser des méthodes différentes, éventuellement moins exactes mais beaucoup plus rapides.

BLAST et la recherche de séquences dans une banque

BLAST est un logiciel[5] qui permet de trouver dans une banque les séquences proche d'une séquence soumise grâce à une approche heuristique [KA1]. Son vocabulaire de base est anglo-saxon. Notamment, son nom est l'acronyme de sa fonction: « *Basic Local Alignment Search Tool* », c'est donc un outil de recherche d'alignements locaux basique... et très populaire.

L'algorithme initial de BLAST effectue très rapidement une recherche d'alignement local sans brèche et évalue le score obtenu par une valeur statistique [KA2]. Il a subi diverses adaptations.

Les principaux mots clefs à connaître pour son emploi sont:

- *Query* = la séquence « requête »,
- *Database* = la banque interrogée, qui est l'ensemble de séquences cibles,
- *Subject* = toute séquence tirée de la banque de données (sujet de la population de cibles),
- *Hit* = similarité significative identifiée entre la séquence requête et une séquence cible,
- HSP ou *High Scoring Pairs* = paire de séquences présentant un forte similarité,
- E-Value ou *Expect value* = nombre d'occurrences d'un *hit* du fait du « pur hasard ».

Principe

A la base, le principe de BLAST repose sur l'idée que les « bons » alignements doivent contenir des petits segments strictement identiques ou de très grande similarité.

Partant d'une courte longueur de mot donnée, w=2 ou 3 pour les protéines (7 ou 11 pour les acides nucléiques), il découpe la séquence requête en mots de longueur w (« hachage »).

Pour chaque mot m ainsi extrait de la requête, il dresse la liste *a priori* des mots de longueur w voisins du mot m qu'il est possible de former, c'est-à-dire de tous les mots possibles de longueur w qui égalent ou dépassent un score S_{voisin} donné de similarité avec le mot m. Si w=3, avec 20 acides aminés, il est possible de former 20^3=8000 mots différents; choisir un score de similarité S_{voisin} élevé permet de ne retenir que quelques uns d'entre eux.

Ensuite, chacun de ces mots voisins est confronté à tous les mots de longueur w de toutes les séquences de la banque de données. Chaque fois qu'une identité est trouvée, un *hit* est enregistré. Comme la partie coûteuse en temps de l'opération est la consultation de la grande masse de données dans la banque, le fait d'avoir constitué une « relativement » courte liste de mots avant « d'attaquer » la banque permet de réduire son interrogation à la seule recherche d'identités de mots courts qui peut être faite très rapidement.

Enfin, pour chaque *hit*, le programme effectue une extension sans brèche de l'alignement dans les deux

5 En fait, une famille de programmes applicables à des séquences de nucléotides ou d'acides aminés, ou à des comparaisons croisées.

sens autour des mots courts identiques. L'extension s'arrête quand elle fait baisser le score d'une quantité supérieure à une valeur X_{stop} fixé. Le score de l'alignement étendu obtenu est alors comparé à un seuil S_{HSP} donné et l'alignement est retenu comme un « alignement de haute similarité » (*High Scoring pair*) si son score excède ce seuil.

Le paramètre E (« *Expect value* ») est une mesure statistique du nombre de *hits* auxquels on peut s'attendre du fait du simple hasard pour une recherche dans une banque de taille donnée. Il rend ainsi compte du « bruit de fond » qui affecte les recherches d'alignements. Il constitue donc un moyen commode de fixer un seuil pour filtrer les résultats et ne retenir que les alignements significatifs. Sa valeur décroît exponentiellement en fonction du score requis pour les alignements (l'obtention d'un alignement de score élevé a très peu de chances d'être le fait du simple hasard). Lorsque sa valeur de filtrage augmente, la liste des résultats jugés significatifs s'allonge en incorporant des résultats de score faible.

L'expression de ce paramètre est

$$E = m.N.K.e^{-\lambda.S}$$

où m est la longueur de la séquence requête, N la taille de la banque (en nombre de nucléotides ou d'acides aminés), K et λ deux constantes statistiquement déterminées en fonction des critères de similarité des alignements [KA1].

Exemple

Séquence requête de LQ=12 acides aminés : *Query* = NKCKTPQGQRLV.

Mots de w=3 lettres extraits de la requête (il y en a LQ-(w-1)=10) :	Pour chaque mot extrait, mots voisins avec Seuil S_{voisin}=13 : cas de PQG	Explication du calcul du score d'après BLOSUM62	Pour chaque mot voisin retenu, recherche de *hits* : cas de PMG
NKC KCK CKT KTP TPQ PQG➔➔➔➔➔ QGQ GQR QRL RLV	Candidat score **PQG** 18 **PEG** 15 **PRG** 14 **PKG** 14 **PNG** 13 **PDG** 13 **PHG** 13 **PMG** 13 **PSG** 13 --- PQA 12 ...	PP=7 + QQ=5 + GG=6 PP=7 + QE=2 + GG=6 PP=7 + QR=1 + GG=6 PP=7 + QK=1 + GG=6 PP=7 + QN=0 + GG=6 PP=7 + QD=0 + GG=6 PP=7 + QH=0 + GG=6 PP=7 + QM=0 + GG=6 PP=7 + QS=0 + GG=6 --- PP=7 + QQ=5 + GA=0 ...	➔Subj: **VCTVTPMGSADL** Nota : la séquence ci-dessus n'est pas réelle, juste un exemple théorique.

A partir du mot de w=3 lettres à l'origine du *hit*, soit PMG qui réalise un score de 13, l'alignement est étendu. Si la valeur d'arrêt X_{stop} est fixée à 3, la progression s'arrête dès que l'extension fait tomber le score décroissant (flèches) en dessous de 10.

Requête	N	K	C	K	T	P	Q	G	Q	R	L	V
BLOSUM62 paires	-3	-3	-1	-2	5	7	0	6	0	-1	-4	1
Hit étendu	V	C	T	V	T	P	M	G	S	A	D	L
BLOSUM62 cumuls	9	12	15	16	18		13		13	12	8	9

L'alignement local obtenu est retenu si son score total (ici 11 de KC à QR) excède le seuil S_{HSP}.

Pour une extension sans brèche évaluée à l'aide de BLOSUM62, comme ci-dessus, les paramètres de l'E-value sont K=0,318 et λ=0,13. La longueur de la requête étant de 12, pour une valeur de score d'au moins 11 (comme celle obtenue), en supposant que la taille de la banque est N=1000, l'E-value vaut 913, ce qui signifie que ce résultat n'est pas significatif.

Alignement multiple

Le but de l'alignement multiple (en anglais « *multiple sequence alignment* » ou MSA) est de mettre en évidence une région commune à tout un groupe de séquences. Comme celui deux à deux, l'alignement multiple peut être global ou local.

Pour le réaliser, il faut disposer d'un système de score permettant de juger de la similarité des séquences, quel que soit leur ordre, et qui puisse être calculé quel que soit leur nombre.

Programmation dynamique

Dans le principe, rien n'interdit d'imaginer une méthode d'alignement global ou local par programmation dynamique exacte, en remplissant une matrice multidimensionnelle de valeurs de similarité obtenues de proche en proche. Cependant, pour remplir chaque cellule de la matrice, le nombre de cellules voisines d'origine à considérer augmente avec le nombre N de séquences. En effet, le nombre de dimensions de la matrice est égal au nombre N de séquences et le nombre de cases d'origine possibles est 2^N-1; il croît donc vite avec N: 3 pour 2 séquences, 7 pour 3, 15 pour 4, ..., 255 pour 8 séquences, etc. Par ailleurs, le nombre de cellules de la matrice est le produit des longueurs de toutes les séquences. Si elles ont la même longueur L, c'est L^N. Sans compter l'initialisation de la matrice, qui est rapide, le nombre de calculs élémentaires de valeurs à comparer, cellule par cellule, pour calculer dynamiquement la similarité de l'ensemble des séquences est donc $L^N.(2^N-1)$. Lorsque N est grand, 2N est très grand devant 1, donc le terme « -1 » peut être négligé dans la parenthèse est l'ordre de grandeur du nombre de calculs élémentaires est donc $L^N 2^N=(2L)^N$.

Ainsi si T_2 et T_N sont les durées de calcul nécessaires pour obtenir respectivement la similarité de 2 ou de N séquences, le rapport entre ces deux durées est approximativement:

$$\frac{T_N}{T_2} = \frac{(2L)^N}{L^2.(2^2-1)} = \frac{(2L)^N}{3L^2}$$

Exemple: pour des protéines de 150 acides aminés (L=150), le temps de calcul de la similarité de N séquences est $T_N = 300^N/67500$. Si, pour fixer les idées, le temps de calcul pour 2 séquences est de une seconde (T_2=1s), alors pour 3 séquences il faut 6,7 min, pour 5 séquences près de 1,1 an et pour 7 environ mille siècles[6].

Comme indiqué plus haut, le temps de calcul d'une résolution exacte est donc rédhibitoire, même si l'on gagnait un facteur 100 sur la puissance des ordinateurs.

Une première direction de recherche pour diminuer ce temps est de trouver des moyens de réduire la portion de la matrice dans laquelle le chemin optimal est recherché en « coupant les coins » de la matrice grâce à des estimations approximatives, ce qui réduit le nombre de scores à calculer sans affecter la résolution exacte de la matrice (si les coins sont convenablement coupés!).

D'autres stratégies existent. Par exemple DCA (de l'anglais « ***D**ivide – and – **C**onquer multiple sequence **A**lignment* ») est un programme qui produit rapidement des alignements de haute qualité. Ce logiciel repose sur une technique simple. L'idée consiste à couper chaque séquence en deux à une position de coupure judicieuse proche de son milieu. Ainsi, au lieu d'une famille de longues séquences, il s'agit d'aligner deux familles de séquences deux fois plus courtes. Le processus est répété jusqu'à ce que les séquences aient une taille qui permette de procéder efficacement à leur alignement multiple. Ensuite, les alignements obtenus sont concaténés pour restituer un alignement multiple des séquences d'origine [S1].

Toutefois, la voie la plus exploitée aujourd'hui s'appuie sur le développement d'algorithmes dits « heuristiques » qui réduisent le nombre de calculs en effectuant une résolution approximative du problème.

6 Le cours Pasteur donne des valeurs différentes parce qu'il omet le facteur 2 (associé à l'augmentation du nombre de cellules d'origine à prendre en compte pour le calcul de chaque case) dans l'expression du temps de calcul. Il sous-estime donc largement l'augmentation du temps de calcul.

Algorithmes heuristiques

Méthode de l'alignement de l'arbre (« tree alignment »)

Cette méthode est utilisable si les séquences peuvent préalablement être rangées selon un arbre phylogénétique. Ce n'est en général pas le cas et ceci oblige à construire l'arbre en même temps que l'alignement.

Alignement progressif: CLUSTAL

CLUSTAL (« ***clust****er* ***al****ignment* ») est l'un des plus populaires algorithmes d'alignement multiple. Deux variantes existent: ClustalV (qui effectue un alignement global) et ClustalW (alignement local, plus récent). Il utilise la procédure suivante :

1. alignement par paires : calcul des scores de chaque paire de séquences, pour toutes les paires possibles;

2. création d'une matrice de distances entre les séquences (utilisant les scores de similarités calculés à l'étape précédente) et identification de groupes de séquences sur la base de cette matrice (plusieurs méthodes existent pour ce faire, la plus utilisée aujourd'hui est celle dite « du plus proche voisin » (« *Neighbor Joining* »);

3. alignement progressif de tous les groupes (« clusters ») en exploitant leur ordre de branchement dans la matrice de similarité calculée précédemment.

La première étape, d'alignement par paires, peut être effectuée dans CLUSTAL par deux méthodes: l'une rapide (« FAST »), l'autre plus lente (« SLOW » !). « FAST » calcule le score comme le nombre de sous-séquences de longueur k identiques dans les deux séquences (k=1 à 2 pour les protéines, 2 à 4 pour les acides nucléiques) diminué d'une pénalité fixée pour chaque brèche. « SLOW » effectue un alignement exact des deux séquences par programmation dynamique et le score est le nombre d'identités trouvées entre les deux séquences divisé par le nombre de résidus comparés (sans compter les insertions ni les délétions).

A l'étape suivante, le score de similarité calculé à l'étape précédente est converti en une distance[7] qui indique le nombre de différences par site (pas de correction pour les substitutions multiples dans ces distances initiales). Les distances obtenues pour toutes les paires sont enregistrées dans la matrice de distances. Cette matrice est ensuite utilisée pour placer les séquences dans un « arbre guide » (qui n'a pas nécessairement une signification phylogénétique[8]). Plusieurs algorithmes existent pour construire cet arbre guide mais le plus connu est celui dû à Saitou et Nei [SN1] qui repose sur la méthode du plus proche voisin, selon la description simplifiée suivante.

Au départ, chaque séquence S_i constitue un « nœud » initial. Pour construire l'arbre guide, l'algorithme cherche d'abord la plus petite valeur de distance dans la matrice et ceci lui désigne les deux séquences, disons S_a et S_b, qui seront les deux « feuilles » de l'arbre les plus voisines (une « feuille » est un nœud extérieur). Ces deux feuilles sont remplacées par un nœud unique S_{ab} et la matrice des distances est mise à jour en calculant la distance de ce nouveau nœud à chacun des autres S_i comme la moyenne des distances de S_a et S_b à S_i: $D(S_{ab}, S_i)= [D(S_a, S_i)+D(S_b, S_i)]/2$. Le processus est répété jusqu'à épuisement des séquences. Chaque fois qu'une paire de nœuds est remplacée, l'algorithme calcule les longueurs de branches reliant chacun des deux nœuds à l'embranchement de façon à rendre compte des distances entre séquences en minimisant la longueur totale des branches de l'arbre achevé. La dernière opération est de placer la racine de l'arbre au point moyen qui rend égales les longueurs de branches de part et d'autre de la racine.

7 Rappel: une distance d indique un éloignement entre deux objets: d grand implique que les deux objets sont très différents. Une distance est une quantité mathématique qui doit respecter certaines contraintes (par définition). Un indice de similarité traduit au contraire la ressemblance entre deux objets: plus il est grand, plus les objets se ressemblent. Si d est une distance, 1-d ou 1/(1+d) sont deux indices de similarité possibles correspondants; on peut en construire d'autres.

8 Car la distance choisie n'a pas nécessairement une relation avec l'intervalle évolutif entre séquences.

La troisième étape est l'alignement progressif réalisé en suivant l'ordre indiqué par la construction de l'arbre guide. Ceci est fait en utilisant un algorithme de programmation dynamique un peu étendu pour permettre l'alignement d'une séquence avec un groupe de séquences déjà alignées ou de deux tels groupes entre eux. Des brèches sont introduites pour optimiser l'alignement et ne sont jamais retirées[9].

Une fois l'alignement terminé, CLUSTAL attribue un poids à chaque séquence. Ce poids est simplement la somme des poids des branches de la racine jusqu'au taxon considéré, où le poids d'une branche est sa longueur divisée par le nombre de taxons qu'elle porte.

Les inconvénients de CLUSTAL

La méthode ne revenant jamais en arrière les erreurs introduites dans les alignements des sous-groupes initiaux se propagent dans tous l'alignement et ne sont jamais corrigés.

Par ailleurs, les matrices de similitude sont adaptées au fil de l'algorithme en fonction de la divergence des séquences à aligner. L'utilisateur ne choisit donc pas pas sa matrice.

La pénalité de brèche est réduite dans les régions hydrophobes.

CLUSTAL encourage la formation de brèche dans les boucles plutôt que dans les régions structurées.

La pénalité de brèche est diminuée près d'autres brèches. Ceci privilégie la formation de longues brèches, au détriment de petits brèches voisines.

9 Les Anglo-saxons caractérisent cette procédure par la formule « *once a gap, always a gap* », qui pourrait être traduite en français par « une brèche un jour, une brèche toujours »

Références

[HH1] Henikoff S., Henikoff J.G., « *Amino acid substitution matrices from protein blocks* », PNAS November 15, vol. 89 no. 22 10915-10919, 1992.

[MOD 1] M.O. Dayhoff, ed., « *Atlas of Protein Sequence and Structure* », Vol5, 1978.

[NW 1] Needleman, S. B., Wunsch, C. D., « *A general method applicable to the search for similarities in the amino acid sequence of two proteins* », J. Mol. Biol., 443-53, 1970.

[SW 1] Smith TF, Waterman MS « *Identification of Common Molecular Subsequences* », J. of Mol. Biol. 147: 195–197, 1981.

[JTT1] Jones, D.T., Taylor, W.R. & Thornton, J.M., « *A new approach to protein fold recognition. Nature* », 358, 86-89, 1992.

[KA1] S. Karlin,. F. Altschul, « *Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes* », Proc. Natl. Acad. Sci. U S A 87: 2264-8, 1990.

[KA2] Karlin, S, SF Altschul, « *Applications and statistics for multiple high-scoring segments in molecular sequences* », Proc. Natl. Acad. Sci. 90:5873-7, 1993.

[MH1] M. Höchsmann, « The Tree Alignment Model: Algorithms, Implementations and Applications for the Analysis of RNA Secondary Structures », thèse soutenue à la faculté technique de l'Université de Bielefeld, Mars 2005.

[S1] J. Stoye, « *Multiple Sequence Alignment with the Divide-and-Conquer Method* », Gene 211, GC45-GC56, 1998.

[SN1] Saitou, N., M. Nei. « *The neighbor-joining method: a new method for reconstructing phylogenetic trees* », Mol. Biol. Evol. 4:406-425, 1987.