Debrief du vendredi 20/01

1 Récapitulatif de la journée

Nous avons dégagé plusieurs pistes de recherche avec G. Fertin et G. Jean :

- il serait intéressant d'analyser les performances d'une première version de la méthode séquentielle décrite dans le debrief du vendredi dernier. Cette première version sera naïve et écartera tout baitModel trop ambigu. Éventuellement, cette version préliminaire ne renverra pas de solution (si tous les baitModels considérés sont ambigus alors ils seront tous ignorés). Cette implémentation sera à tester sur les 10 000 baits fournis dans le fichier de données. Nous cherchons surtout à savoir pour combien de baits nous arrivons à obtenir un baitModel "fusionné". Lorsqu'un tel baitModel est obtenu nous quantifierons sa proximité avec le bait.
- un premier état de l'art sur les méthodes d'alignement de séquences multiples (ou MSA pour "Multiple Sequence Alignment") doit être dressé pour discuter de la viabilité de la méthode de fusion de baitModels utilisant l'alignement de séquences (voir debrief du vendredi 13/01). De plus, des tests utilisant au moins une méthode de la littérature sont attendus pour déterminer comment étendre ces méthodes à notre problème.

Dans ce debrief, nous présenterons tout d'abord un état de l'art des méthodes MSA pour ensuite interpréter les résultats obtenus avec ClustalX (version graphique de ClustalW) sur quelques instances.

2 Alignement de séquences

Il existe dans la littérature plusieurs approches pour résoudre le problème d'alignement. Cependant, ce problème est NP-difficile et la complexité des algorithmes pour résoudre ce problème à l'optimal est souvent exponentielle. Nous présenterons dans un premier temps quelques types d'alignement existant. Dans un second temps nous présenterons les algorithmes d'alignement pair-à-pair. Enfin, étant donné que nous manipulons parfois plus de deux baitModels, nous présenterons les algorithmes d'alignement multiple.

2.1 Types d'alignement

2.1.1 Alignement global

L'alignement global consiste à aligner l'intégralité de deux (ou plus) de séquences. En d'autres mots, les séquences sont alignées de sorte à maximiser le nombre de lettres alignées. Cet alignement est adapté aux séquences étroitement liées [1]. Il est plus stable lorsque les sections de forte similarité sont déjà plus ou moins bien alignées.

2.1.2 Alignement local

L'alignement local consiste à aligner des sections ayant une forte similarité. Cette méthode aligne des sections de séquences et est adapté aux séquences ayant peu de similarités entre elles [1]. Cet alignement est plus stable que l'alignement global lorsque les sections de forte similarité sont peu ou pas alignées.

2.1.3 Alignement semi-global

Les différences entre l'alignement global et l'alignement local ainsi que leur avantage respectif ouvre la possibilité de créer une méthode combinée permettant de générer des alignements plus précis [1]. L'alignement semi-global (ou "glocal") est une méthode hybride entre l'alignement global et l'alignement local alignant les séquences de sorte à obtenir le meilleur alignement partiel des séquences [2, 3]. Cet alignement ignore les écarts au début et/ou à la fin d'un alignement. Il est notamment utile si une séquence est beaucoup plus courte qu'une autre (e.g. lors d'une comparaison entre un peptide et un protéome). Dans ce cas, la séquence la plus courte devrait être alignée globalement (dans son entièreté) mais la séquence la plus longue devrait être alignée localement (partiellement).

2.2 Alignement pair-à-pair

Ces méthodes sont utilisées pour évaluer les similarités entre deux séquences. Nous serons amenés à fusionner plus de deux baitModels mais les méthodes pair-à-pair sont utilisées par certaines méthodes d'alignement multiple et font partie des premières méthodes qui ont été développée pour le problème d'alignement. Nous présentons ici l'algorithme de Needleman-Wunsch pour l'alignement global et celui de Smith-Waterman pour l'alignement local. Ces méthodes sont implémentées avec des notions de programmation dynamique.

2.2.1 Algorithme de Needleman-Wunsch

Soient deux séquences S et T. La méthode de Needleman-Wunsch [4, 3] utilise la programmation dynamique pour trouver l'alignement global optimal entre S et T en $O(nm)$. Soit $V(i, j)$ le score de l'alignement optimal entre $S[1 \dots i]$ et $T[1 \dots j]$. Soit $\delta(a, b)$ le score de similarité entre deux lettres a et b. La méthode définie une formule récursive pour $V(i, j)$ selon deux cas de figure : (1) soit $i = 0$ ou $j = 0$; (2) soit $i > 0$ et $j > 0$.

Dans le premier cas, on aligne une séquence avec une séquence vide (nous avons donc des insertions ou suppressions) et l'on a

$$
\begin{aligned}
&V(0,0) = 0 \\
&V(0,j) = V(0,j-1) + \delta(\_, T[j]) \quad \text{Insère } j \text{ fois} \\
&V(i,0) = V(i-1,0) + \delta(S[i], \_) \quad \text{Supprime } i \text{ fois}
\end{aligned}
$$

Dans le deuxième cas, lorsque $i > 0$ et $j > 0$, dans le meilleur alignement entre $S[1 \dots i]$ et $T[1 \dots j]$, la dernière pair de lettres alignées devrait être soit une correspondance entre les lettres, une suppression ou une insertion de lettres. Le score optimal correspondra à la valeur maximum entre ces trois cas, on a :

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + \delta(S[i], T[j]) & \text{correspondance} \\ V(i-1, j) + \delta(S[i], \_) & \text{suppression} \\ V(i, j-1) + \delta(\_, T[j]) & \text{insertion} \end{cases}$$

Le score de l'alignement optimal de $S[1 \dots n]$ et $T[1 \dots m]$ est $V(n, m)$. Pour obtenir ce score, les cases d'une table de taille n par m sont remplies en utilisant les équations récursives présentées ci-dessus. Le score de l'aligment optimal est indiqué dans la dernière case de la table (en bas à droite sur la FIGURE 2). La FIGURE 1 indique les scores de similarité δ entre les lettres des séquences de l'exemple sur la FIGURE 2.

	-	A	C	G	T
-		-1	-1	-1	-1
A	-1	2	-1	-1	-1
C	-1	-1	2	-1	-1
G	-1	-1	-1	2	-1
T	-1	-1	-1	-1	2

FIGURE 1 – Matrice des scores de similarité entre A, C, G et T

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

FIGURE 2 – Table pour la méthode Needleman-Wunsch pour deux séquences $S = ACAATCC$ et $T = AGCATGC$. Le score optimal de l'alignement est ici 7. **Illustration tirée du livre de WK Sung [3].**

Pour retrouver l'alignement optimal, pour chaque case, une flèche noire retrace le chemin emprunté pour arriver à la dernière case de la table. Si la flèche est diagonale, les deux lettres correspondantes sont alignées. Si la flèche est horizontale ou verticale alors il y a une suppression ou une insertion (respectivement) dans l'alignement. En retraçant le chemin de $V(7,7)$ à $V(0,0)$, le chemin est $0 \leftarrow 1 \leftarrow 3 \leftarrow 5 \leftarrow 4 \leftarrow 6 \leftarrow 5 \leftarrow 7$. Plusieurs solutions peuvent être obtenues dont $A - CAATCC$ et $AGCA - TGC$ ici.
La version classique de Needleman-Wunsch nécessite de remplir une table de taille n par m donc l'algorithme est en $O(nm)$ en temps et en espace (étant donné qu'une case se remplie en $O(1)$).

D'autres versions ont été proposées dans la littérature pour améliorer la complexité temporelle ou spatiale de la version de base [5, 6] (la meilleure méthode actuelle en temps est en $O(nm/\log n)$ et la meilleure en espace est en $O(m+n)$).

2.2.2 Algorithme de Smith-Waterman

La méthode de Smith-Waterman [7]

À COMPLÉTER

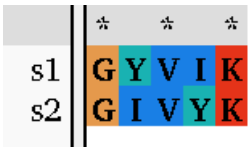
2.3 Alignement multiple

À COMPLÉTER

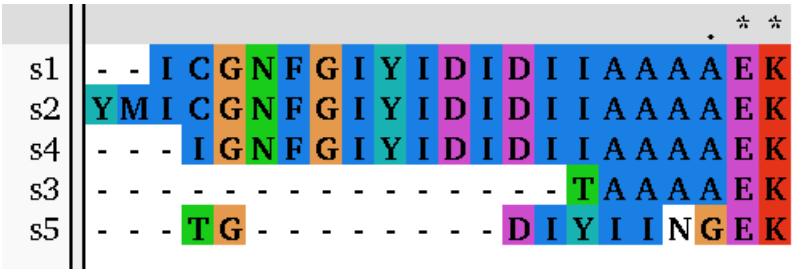
3 Résultats sur ClustalX

Nous avons utiliser ClustalX (version graphique de ClustalW), pour calculer l'alignement multiple de quelques baitModels. Nous avons choisi la matrice PAM 350 pour la matrice des poids et tous les autres paramètres de ClustalX sont laissés à leurs valeur par défaut. Quelques baitModels sont enregistrés au format FASTA dans le répertoire instances/-FASTA et sans tenir compte des crochets.

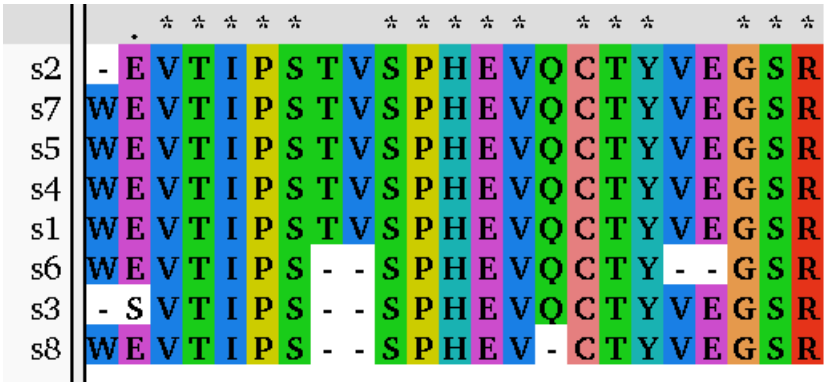
Premièrement, la suppression totale des crochets peut sursimplifier l'alignement. Sur l'instance GTFQIVYK avec les deux baitModels G[89.09]YVI[287.09]K et G[376.17]IVYK, on obtient l'alignement suivant :



De plus, dans la continuité des remarques faites dans le debrief précédent, la masse 89.09 n'existe pas et si la séquence entre crochets (YVI) n'est pas dans "le bon sens" alors ClustalX trouvera plus de mismatchs (ici entre les pairs de I et Y). Cette instance est particulièrement compliquée étant donné qu'il n'y a pas d'informations dans les baitModels pour déduire quels sont les acides aminés manquant. Cependant, cela implique que sur les instances avec beaucoup de baitModels variés, l'alignement multiples pourra éventuellement trouver de bonnes solutions. Exemple sur l'instance TYTYMICGNFGIYIDIDIIAAAAEK :



Autre exemple sur l'instance WEVTIPSTVSPHEVQCTYVEGSR :



Ainsi, ClustalW est une méthode de la littérature bien documentée et potentiellement efficace sur les instances avec beaucoup de baitModels peu ambigus (sur ce genre d'instance il suffirait de sélectionner pour chaque colonne l'acide aminé le plus représenté).

Le désavantage de cette méthode dépendent de deux problèmes :

- certains baitModels apportent des informations contreproductives mais qui se rapprochent d'une information correcte (e.g. YVI au lieu de IVY mais $m(\text{YVI}) = m(\text{IVY})$)
- Lorsque les baitModels sont bien construits, on obtiendra un alignement "correct". Si ensuite on décide de sélectionné chaque acide aminé le plus représenté par colonne, il se peut que deux acides aminés soit représentés de la même façon. Il faut alors trancher entre les deux choix ou proposer les deux (e.g. FIGURE 3).

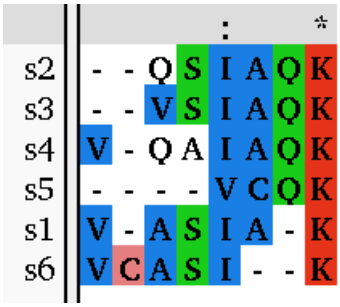


FIGURE 3 – pour l'instance VCASIAQK, en utilisant la méthode décrite, on peut soit sélectionner Q soit sélectionner A comme troisième acide aminé

Références

[1] Valery O Polyanovsky, Mikhail A Roytberg et Vladimir G Tumanyan. « Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences ». In : *Algorithms for molecular biology* 6.1 (2011), p. 1-12.

[2] Michael Brudno et al. « Glocal alignment : finding rearrangements during alignment ». In : *Bioinformatics* 19.suppl_1 (2003), p. i54-i62.

[3] Wing-Kin Sung. *Algorithms in bioinformatics : A practical introduction*. Chapman et Hall/CRC, 2009.

[4] Saul B Needleman et Christian D Wunsch. « A general method applicable to the search for similarities in the amino acid sequence of two proteins ». In : *Journal of molecular biology* 48.3 (1970), p. 443-453.

[5] William J Masek et Michael S Paterson. « A faster algorithm computing string edit distances ». In : *Journal of Computer and System sciences* 20.1 (1980), p. 18-31.

[6] Daniel S. Hirschberg. « A linear space algorithm for computing maximal common subsequences ». In : *Communications of the ACM* 18.6 (1975), p. 341-343.

[7] Temple F Smith, Michael S Waterman et al. « Identification of common molecular subsequences ». In : *Journal of molecular biology* 147.1 (1981), p. 195-197.