



République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

**Université des Sciences et de la Technologie Houari Boumediene**

Faculté d'Informatique  
Département Informatique

**Spécialité : Bio-Informatique**

**Mini-Projet Module Bio-Algorithme**

**Thème**

**Alignement multiple de séquences**

---

**Réalisé par :**

**LAIB Ayoub**

---

# Table de matière

1.	Introduction .....	1
1.1.	BIO-INFORMATIQUE .....	1
1.1.1.	Définition .....	1
1.1.2.	Les Types d'informations .....	2
1.2.	Objectif du Projet .....	2
2.	Notions de base .....	3
2.1.	Séquences biologiques .....	3
2.1.1.	ADN.....	3
2.1.2.	ARN.....	4
2.1.3.	Protéines .....	5
2.2.	Alignement de séquences.....	6
2.2.1.	Données biologiques.....	6
2.2.2.	Alignement.....	7
2.2.3.	Score d'un alignement .....	7
2.2.4.	Pourquoi aligner des séquences .....	8
2.2.5.	Type d'alignements .....	8
2.3.	Matrices de similarité .....	9
2.3.1.	POINT ACCEPTED MUTATION .....	9
2.3.2.	BLOCKS OF AMINO ACID SUBSTITUTION MATRIX.....	11
3.	Alignement par paire de séquences.....	12
3.1.	L'algorithme de Needleman et Wunsch .....	12
3.2.	Exemple d'alignement .....	12
3.3.	Calcul des scores .....	16
3.4.	Utilisation de l'Alignement par Paires de Séquences .....	17
3.4.1.	Comparaison de Gènes .....	17
3.4.2.	Étude de Mutations et de Variantes Génétiques.....	17
3.4.3.	Motivation de l'Utilisation de l'Alignement par Paires.....	17
3.5.	Implémentation .....	18
3.6.	Résultat de l'alignement .....	18
3.7.	Tests sur des séquences de différentes longueurs .....	19
3.8.	Analyse de la complexité expérimentale .....	20
3.8.1.	Temps d'exécution constant pour petites séquences.....	20
3.8.2.	Augmentation du temps d'exécution avec la taille des séquences .....	20
3.8.3.	Comportement linéaire à quadratique .....	20
3.8.4.	Variabilité et précision des mesures .....	20
4.	Alignement multiple de séquences .....	20
4.1.	Exemple d'alignement multiple .....	22

4.2.	<b>Utilisation de l'alignement multiple .....</b>	22
4.3.	<b>Méthodes d'alignement multiple .....</b>	23
4.3.1.	<b>L'Approche Exacte .....</b>	23
4.3.1.1.	<b>La Programmation dynamique.....</b>	24
4.3.1.2.	<b>Avantages .....</b>	24
4.3.1.3.	<b>Limites .....</b>	24
4.3.2.	<b>Méthodes Itérative .....</b>	24
4.3.2.1.	<b>La Méthode DIALIGN .....</b>	25
4.3.2.2.	<b>La Méthode KALIGN .....</b>	25
4.3.2.3.	<b>Avantages .....</b>	26
4.3.2.4.	<b>Limites .....</b>	26
4.3.3.	<b>L'Approche Progressive .....</b>	26
4.3.3.1.	<b>Avantages .....</b>	28
4.3.3.2.	<b>Limites .....</b>	28
4.4.	<b>Amélioration des performances MSA dans le Big Data .....</b>	28
4.5.	<b>Implémentation d'une méthode progressive .....</b>	29
4.5.1.	<b>La Méthode ClustalW.....</b>	29
4.5.1.1.	<b>Nouvelle version : Clustal Omega .....</b>	30
4.5.2.	<b>Notion de profil .....</b>	31
4.5.3.	<b>Affichage les alignements obtenus progressivement .....</b>	31
4.6.	<b>Tests .....</b>	32
4.6.1.	<b>Tests avec la longueur de séquences constante .....</b>	32
4.6.1.1.	<b>Analyse .....</b>	33
4.6.2.	<b>Tests avec différents longueur de séquences .....</b>	33
4.6.2.1.	<b>Analyse .....</b>	34
4.7.	<b>Conclusion .....</b>	34
5.	<b>Autre méthode d'alignement multiple .....</b>	35
5.1.	<b>La Méthode SAGA (Méthodes Itérative).....</b>	35
5.2.	<b>Implémentation .....</b>	35
5.3.	<b>Tests .....</b>	36
5.4.	<b>Analyse.....</b>	37
5.5.	<b>Conclusion .....</b>	37
6.	<b>Comparaison entre les deux méthodes .....</b>	38
6.1.	<b>Temps d'exécution .....</b>	38
6.2.	<b>Qualité de l'alignement .....</b>	39
6.3.	<b>Scalabilité .....</b>	39
7.	<b>Conclusion .....</b>	40
8.	<b>Annexe .....</b>	41

# Table de Figures

Figure 1: Le Paysage Interdisciplinaire de la Bio-informatique.....	1
Figure 2: Types d'informations biologiques.....	2
Figure 3: Exemple des Deux Brins d'ADN .....	3
Figure 4: Représentation des Brins d'ADN et d'ARN et leurs Nucléotides .....	4
Figure 5: De l'ADN à la Protéine : Transcription et Traduction .....	5
Figure 6: Comprendre l'Histoire et l'Évolution des Organismes.....	6
Figure 7: Alignement Multiple de Séquences.....	8
Figure 8: Alignement Multiple de Séquences.....	9
Figure 9: Matrice de Substitution PAM 250 Utilisée pour l'Alignement de Séquences Protéiques .....	10
Figure 10: BLOCKS OF AMINO ACID SUBSTITUTION MATRIX .....	11
Figure 11: Construction de la matrice initiale .....	13
Figure 12: Construction de la matrice transformée .....	13
Figure 13: L'application de l'algorithme de Needleman-Wunsch .....	14
Figure 14: Résultat de l'algorithme de Needleman-Wunsch.....	15
Figure 15: Traceback dans une matrice de scores pour l'alignement de séquences .....	15
Figure 16: Génération de l'alignement.....	16
Figure 17: Résultat de l'exécution du programme d'alignement .....	18
Figure 18: Extrait des résultats d'exécution de l'alignement.....	19
Figure 19: Exemple d'un alignement multiple de séquences.....	21
Figure 20: Calcul d'un score d'alignement multiple .....	22
Figure 21: Calcul d'un score d'alignement multiple - Approche progressif.....	27
Figure 22: Alignement multiple - Approche progressif .....	27
Figure 23: Processus de la méthode ClustalW .....	29
Figure 24: Processus de la méthode Clustal Omega.....	30
Figure 25: Résultat de l'exécution du programme d'alignement .....	31
Figure 26: Extrait des résultats d'exécution de l'alignement - Approche Progressif.....	32
Figure 27: Tableau explicatif de l'alignement - Approche itérative (SAGA) .....	37
Figure 28: Comparaison entre les deux méthodes en fonction de temps exécution .....	38
Figure 29: Tableau explicatif de l'alignement par paire.....	41
Figure 30: Tableaux explicatif de l'alignement - Approche progressif.....	42
Figure 31: Tableaux explicatif de l'alignement - Approche itérative (SAGA) .....	44

## 1. Introduction

### 1.1. BIO-INFORMATIQUE

#### 1.1.1. Définition

Ensemble de méthodes, de logiciels et d'applications en ligne qui permettent de gérer, manipuler, et analyser des données biologiques.

La bio-informatique est une « inter-discipline » à la frontière de la biologie, de l'informatique et des mathématiques qui permet d'analyser les informations biologiques énorme contenues dans les cellules vivantes sous forme de séquences nucléiques ou protéiques.

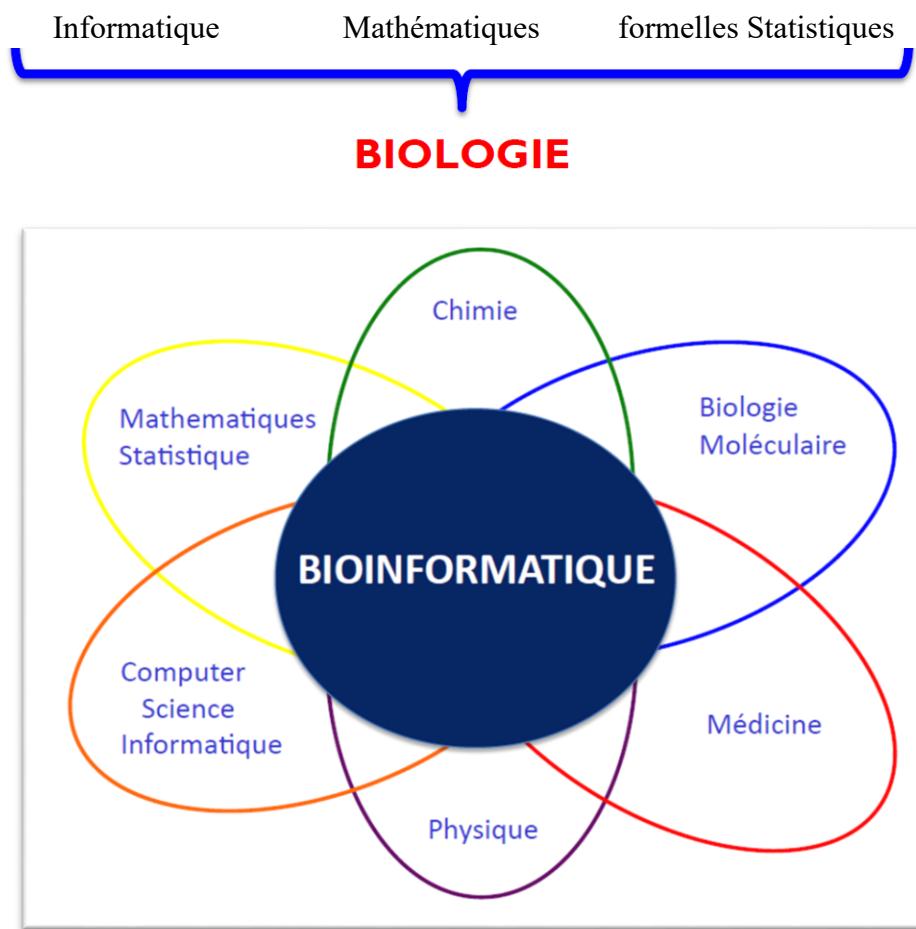


Figure 1: Le Paysage Interdisciplinaire de la Bio-informatique

### 1.1.2. Les Types d'informations

- Génome (l'ensemble du matériel génétique d'un individu ou d'une espèce.)
- Transcriptome (l'ensemble des ARN messagers transcrits à partir du génome)
- Protéome (l'ensemble de protéines exprimées à partir du génome)
- Métabolome (l'ensemble des composés organiques (sucres, lipides, amino-acides, ...))
- Intéractome (l'ensemble des interactions protéine-protéine)...

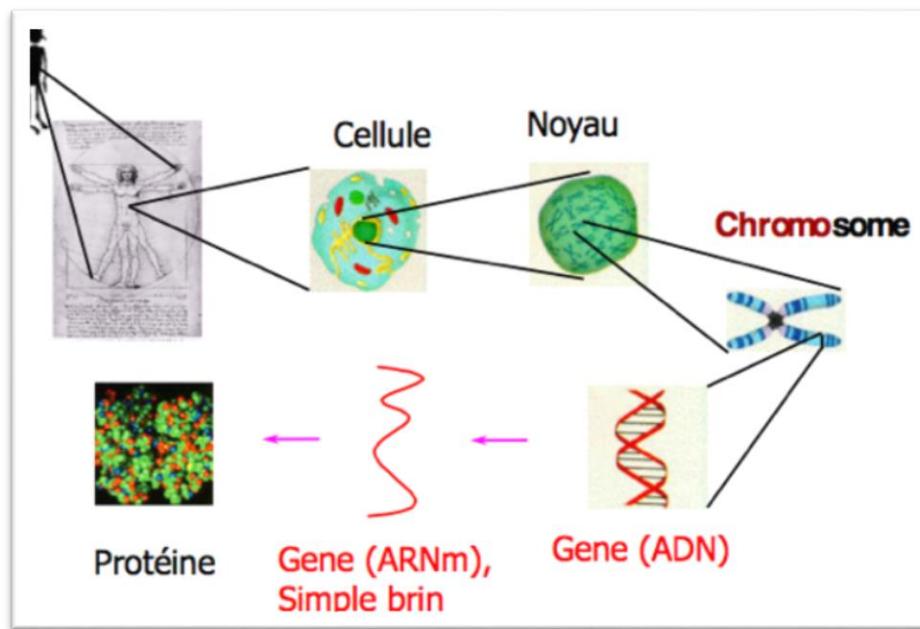


Figure 2: Types d'informations biologiques

### 1.2. Objectif du Projet

L'objectif principal de ce projet est d'explorer et d'analyser les différentes méthodes d'alignement de séquences biologiques, un aspect fondamental de la bio-informatique. L'alignement de séquences permet de comparer les séquences d'ADN, d'ARN, et de protéines pour en déduire des relations évolutives, prédire les fonctions des gènes et des protéines, et identifier des mutations.

Pour atteindre cet objectif, nous allons étudier l'alignement par paire et l'alignement multiple de séquences. Nous commencerons par une explication et une implémentation de l'algorithme de Needleman-Wunsch pour l'alignement global par paire, suivie de tests et d'une analyse de la complexité. Ensuite, nous aborderons l'alignement multiple de séquences en explorant différentes approches, notamment progressive et itérative, en les implémentant et en évaluant leurs performances à travers divers tests. Enfin, nous comparerons les méthodes utilisées pour comprendre leurs avantages, leurs limites, et leur efficacité.

## 2. Notions de base

### 2.1. Séquences biologiques

#### 2.1.1. ADN

L'ADN est le support de l'information génétique.

L'ADN est une longue molécule, faite de deux brins s'enroulant en une double hélice. Les deux brins de la double hélice suggèrent un mécanisme de réPLICATION de l'ADN. Chaque brin est le support d'une succession de nucléotides.

Quatre types de nucléotides: (Adénine, Cytosine, Guanine, Thymine).

Le texte génomique est écrit dans un alphabet de 4 lettres : A, C, G, T

La structure de l'ADN représenté par la figure 2, les deux brins sont orientés, on parle de l'orientation 5' → 3' par des considérations biochimiques.

L'extrémité 5' se termine par un groupement phosphate.

L'extrémité 3' se termine par un groupement hydroxyle.

Il faut retenir qu'il y a un brin sens orienté 5' → 3' et un brin anti-sens orienté 3' → 5', et généralement lorsqu'on veut connaître la séquence des nucléotides le long d'un brin, on va la lire traditionnellement dans le sens 5' → 3'.



Figure 3: Exemple des Deux Brins d'ADN

## 2.1.2. ARN

L'ARN (acide ribonucléique) joue un rôle crucial dans l'expression des gènes et la synthèse des protéines. Contrairement à l'ADN, l'ARN est généralement une molécule simple brin, bien qu'il puisse former des structures secondaires complexes par appariement de bases intramoléculaires.

L'ARN est composé de quatre types de nucléotides, similaires à ceux de l'ADN, à une exception près :

- Adénine (A)
- Cytosine (C)
- Guanine (G)
- Uracile (U) (remplaçant la Thymine de l'ADN)

Le texte génomique de l'ARN est donc écrit dans un alphabet de 4 lettres : A, C, G, U.

La structure de l'ARN est également orientée de manière similaire à celle de l'ADN, avec une orientation 5' → 3' due à des considérations biochimiques :

- L'extrémité 5' de l'ARN se termine par un groupement phosphate.
- L'extrémité 3' se termine par un groupement hydroxyle.

Lorsqu'on veut connaître la séquence des nucléotides le long d'un brin d'ARN, on la lit traditionnellement dans le sens 5' → 3'. Cette orientation est essentielle pour la synthèse des protéines, car les ribosomes lisent l'ARN messager (ARNm) dans ce sens pendant la traduction.

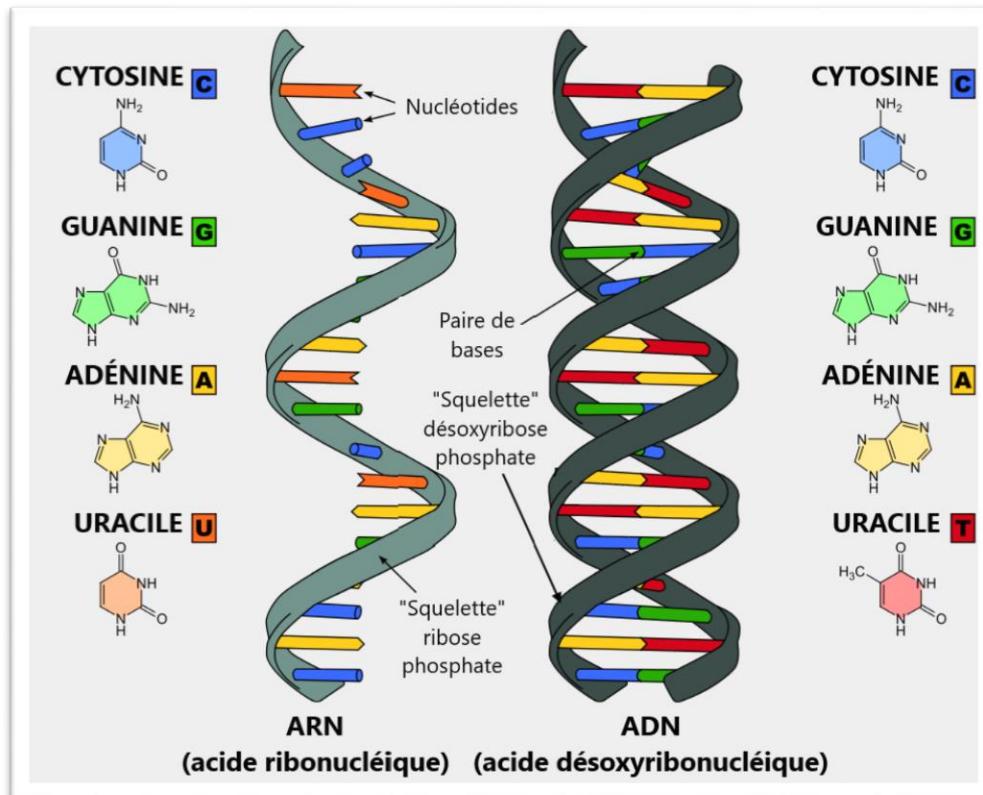


Figure 4: Représentation des Brins d'ADN et d'ARN et leurs Nucléotides

### 2.1.3. Protéines

Les protéines sont des macromolécules essentielles à la vie, impliquées dans presque toutes les fonctions cellulaires. Elles sont composées de chaînes d'acides aminés, qui se replient en structures tridimensionnelles nécessaires à leur fonction.

Les protéines sont constituées de 20 types différents d'acides aminés. Chaque acide aminé est codé par une séquence de trois nucléotides dans l'ARN messager, appelée codon. Les 20 acides aminés sont :

Alanine (Ala, A), Arginine (Arg, R), Asparagine (Asn, N), Aspartate (Asp, D), Cystéine (Cys, C), Glutamate (Glu, E), Glutamine (Gln, Q), Glycine (Gly, G), Histidine (His, H), Isoleucine (Ile, I), Leucine (Leu, L), Lysine (Lys, K), Méthionine (Met, M), Phénylalanine (Phe, F), Proline (Pro, P), Sérine (Ser, S), Thréonine (Thr, T), Tryptophane (Trp, W), Tyrosine (Tyr, Y), Valine (Val, V).

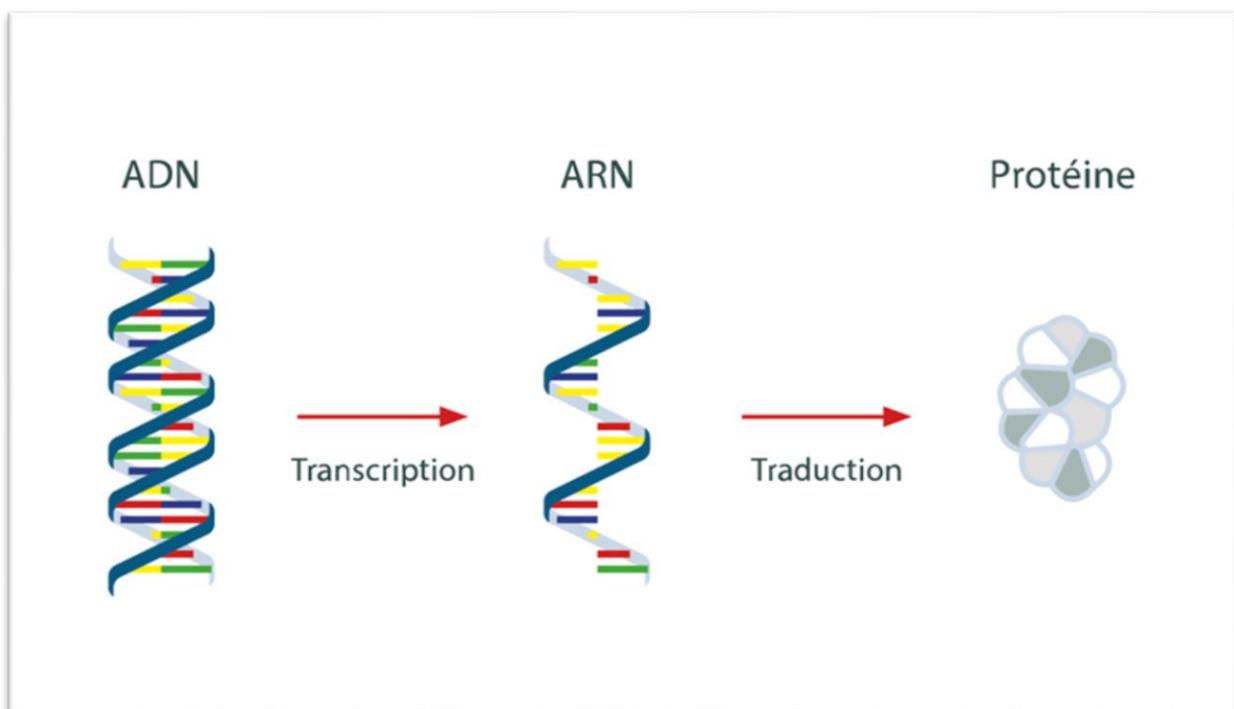


Figure 5: De l'ADN à la Protéine : Transcription et Traduction

## 2.2. Alignement de séquences

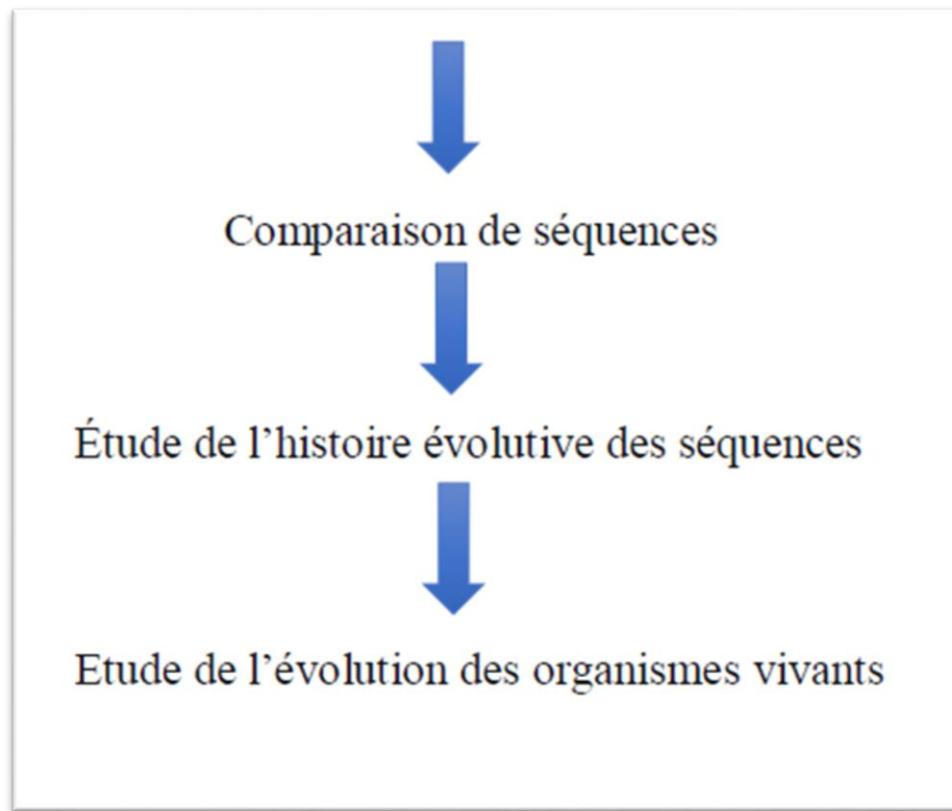
### 2.2.1. Données biologiques

Séquence génomique = suite de lettres - Séquence nucléotidique (ADN) : 4 acides nucléiques

ATGAAGGCTCCCACCGTGCTGGCACCTGGCATTCTGGTGCTGCTGCTTGCCTTG- -

Séquence protéique (protéine) : 20 AA

MKAPTVLAPGILVLLSLVQRSHGECKEALVKSEMNVNMKYQLPNFTAET



*Figure 6: Comprendre l'Histoire et l'Évolution des Organismes*

## 2.2.2. Alignement

Mise en correspondance de deux séquences (ADN ou Protéines) pour faire apparaître les similarités, i.e., segments communs.

Aligner deux séquences globalement c'est les réécrire :

- On rajoute des caractères de gap “-”.
- Elles font la même longueur.
- On ne met pas de gap en face l'un de l'autre.

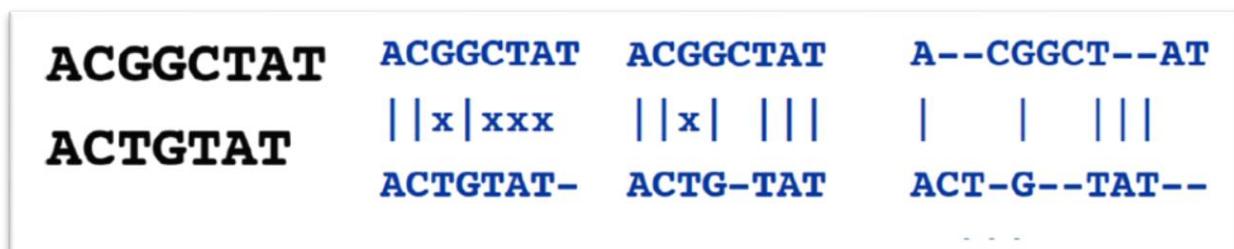


## 2.2.3. Score d'un alignement

La plupart des méthodes d'alignement de séquences biologiques, et en particulier les méthodes d'alignement de séquence de protéines cherchent à optimiser un score d'alignement. Ce score est relié au taux de similarité entre les deux séquences comparées. Il tient compte d'une part du nombre d'acide aminés identiques entre les deux séquences et d'autre part du nombre d'acides aminés similaires sur le plan physico-chimique.

On doit ordonner les qualités avec un score et on somme 3 événements élémentaires le long de l'alignement :

- Même lettre : match
- Lettre différente : mismatch
- Insertion/Délétion (indel)



## 2.2.4. Pourquoi aligner des séquences

L'objectif de l'alignement est de disposer les composants (nucléotides ou aa) et évaluation de la ressemblance globale entre deux séquences pour identifier les zones de concordance. Ces alignements sont réalisés par des programmes informatiques dont l'objectif est de maximiser le nombre de coïncidences entre nucléotides ou acides aminés dans les différentes séquences.

## 2.2.5. Type d'alignements

On distingue 2 types d'alignements qui diffèrent suivant leur complexité :

**L'alignement par paires** : consiste à aligner 2 séquences peut être réalisé grâce à un algorithme de complexité polynomiale. Il est possible de réaliser un alignement :

**Global**, c'est à dire entre les 2 séquences sur toutes leurs longueurs

**Local** entre une séquence et une partie de l'autre séquence

metL	16 KFGGSSLADVKCYLRVAGIMAEYSQPDDMMVVSAAGSTTNQLINWLK-LS	64
	: :  ..... .  :....::   :  :.... . :.... .	
lysC	8 KFGGTSAVDFDAMNRSADIVLSDANV-RLVVLASAGITNLLVALAEGLE	56

**L'alignement multiple**, qui est un alignement global : consiste à aligner plus de 2 séquences et nécessite un temps de calcul et un espace de stockage exponentiel en fonction de la taille des données.

	10	20	30	40	50	60	70
1 GNLSPA1A3	MAELTIDPTTIRKALDEFVESYKPCDTPTQEVGYVATACDGIAHVTCGLPGCMANELLTFEDG--	TLGLAFNLDA					
2 GNLSPA0L5	MSELTITRPRETRAALREFVSSYTPDVASREEVGRVTEAGDGIAIRIEGLPSTMANELLRFEDG--	TLGLAALNLDV					
3 GNLSPA0R2	MAELTISAADIEGAIEDYVSSFSAD-TEREIIGTVIDAGDGIAHVEGLPSVMTQELLEFFGG--	VLGVVALNLDE					
4 GNLSPA0PU	MAELTISANDIQSAIEEYVGSGFTSD-TSREEVGTVDAGDGIAHVEGLPSVMTQELLEFFGG--	VLGVVALNLDE					
5 GNLSPA0QC	MAELTISADDIQSAIEEYVGSGFTSD-TSREEVGTVDAGDGIAHVEGLPSVMTQELLEFFGG--	VLGVVALNLDE					
6 GNLSPA1E9	--MATLRRVDEINKILRERIEQYNRK-VGIENIGRVRVQVGDDIARIIGLGEIMSGELVERFAEG--	TRGIALNLES					
7 GNLSPA1EA	--MATLRRVDEINKILRERIEQYNRK-VGIENIGRVRVQVGDDIARIIGLGEIMSGELVERFAEG--	TRGIALNLES					
8 GNLSPA0A3	--MVTIRADEISNIIRERIEQYNRE-VKIVNTCTVLQVGDDIARIHGLDEVMAGELVEFEEG--	TIGIALNLES					
9 GNLSPA0ZZ	--MVTIRADEISNIIRERIEQYNRE-VKIVNTCTVLQVGDDIARIHGLDEVMAGELVEFEEG--	TIGIALNLES					
10 GNLSPA0T0	--MINIRPDEIISSIIREQIEKYDQQ-VKVNIDGTVLQVGDDIARVYGLDQVMGELLEFEDK--	TIGIALNLEN					
11 GNLSPA0T0	--MINIRPDEIISSIIREQIEKYDQQ-VKVNIDGTVLQVGDDIARVYGLDQVMGELLEFEDK--	TIGIALNLEN					
12 GNLSPA1AX	--MQLNAHEISDLIKKQIIEGFDFD-AEVERTEGSVSVSVDGIVRIHGLADVFQGEMMLEFPNN--	TFGMALNLBQ					
13 GNLSPA0L2	--MQLNSTEISDLIKQRIEQFEVV-SESNEGTTIVASVDGIIIRHGLADVMQGEMIRLPGS--	RFAIALNLER					
14 GNLSPA1JT	--MQLNSTEISDLIKQRIEQFNVV-SEARNNEGTTIVASVDGIIIRVHGLADVMQGEMIRLPGN--	RYAIALNLER					
15 GNLSPA0Q8	--MQLSPSEISGLIKQRIEKFDNS-VELKSEGTTIVSVDGIVTIYGLNDVAAGEMIKLPGD--	VYGLALNLNT					
16 GNLSPA0K2	--MQLNPSETSEISLKSRIQNLQEA-ADVRNQGTIVSVDGIVRIHGLSDVMQGEMLEFFGN--	TFGLALNLER					
17 GNLSPA1K1	--MQLNPSETSEISLKSRIQNLQEA-ATSERNEGTVVSVDGIVTRIHGLTDVMQGEMLEFFGN--	TFGLALNLER					
18 GNLSPA0LL	--MEIRAEISQIIREQIKDYEQ-VELSETGRVLSVGDDIARVYGVKECMSMELLEFFPTEHGVVYGLALNLLEE						
19 GNLSPA0Q2	--MNVPKEEITSIILKKQIIESYEHK-IQTVDSGIIQIGDGIAVYGGIEDCMEGELLEFFPND--	VYGMALNLBQ					
20 GNLSPA0RL	--MSIRABEISALIKQQIENYQSE-IEVSDVGTIVQVGDDIARAHGLDNVMAGELVEFSGN--	VMGLAQNLLEE					
21 GNLSPA0RR	--MSVKLKADEISIILKERIENYNLs-VDIETGTIVSADGVANVYGLKNVMAGEMVFEFTG--	EKGMLALNLEE					
22 GNLSPA1BJ	--MSTTVRPD8VSSILRKQLAGFSESE-ADVYDVGTVLQVGDDIARVYGLSKAAAGELLEFFPNK--	VMGMALNLEE					
23 GNLSPA0LD	--MQVSVAEISGILKKQIAEYK8-ABVSEVGPEVIAVGDDIARAYGLDNVMAGEMVBFEDG--	TQGMALNLBQ					
24 GNLSPA1B8	--MCIQAAEISAILKDQIKNFGQD-AEVAEVGQVLSQLVGDDIARVYGLDKVQAGEMVEFPCC--	IRGMVLNLLET					

Figure 7: Alignement Multiple de Séquences

## 2.3. Matrices de similarité

En pratique, l'évolution a pu faire disparaître (délétion) ou apparaître (insertion) des acides aminés à l'intérieur des séquences. Une délétion dans une séquence correspond à une insertion dans l'autre.

**Indel** : est un mot-clé inventé par le mathématicien Joseph Kruskal et utilisé en génétique et en bio-informatique pour désigner une insertion ou une délétion dans une séquence biologique (acide nucléique ou protéine) par rapport à une séquence de référence.

Une **matrice de substitution** permet, pour chaque acide aminé, de connaître sa capacité à être substitué par chaque autre acide aminé, y compris lui-même.

Les deux types de matrice utilisent des scores basés sur la comparaison entre la fréquence observée des substitutions et leur fréquence attendue.

### 2.3.1. POINT ACCEPTED MUTATION

Ce type de matrice donne la probabilité que, suite à une mutation par substitution au cours de l'évolution, n'importe quel acide aminé remplace n'importe quel autre acide aminé sans que la fonction de la protéine ne soit altérée, d'où la terminologie "mutation acceptée".

metL	16 KFGGSSLADVKCYLRVAGIMAEYSQFDMMVVSAAGSTTNQLINWLK-LS     :: :  ..... .  :.... .::  :  :   ..  .:  .. .	64
lysC	8 KFGGTSAVDFDAMNRSADIVLSDANV-RLVVLASAGITNLLVALAEGLE	56
metL	65 QTDRILSAHQQQTLRRYQCDLISGL---LPAEADSLISAFVSDLERLA .:: .. .:... ... ... ...  .. . ... ... ... ...	110
lysC	57 PGERF---EKLDAIRNIQFAILERLRYPNVIREEIERLLEN-ITVLAEEAA	102
metL	111 ALLDGGINDAVYAEVVGHGEVWSARLMSAVLNQQGLPAAWLDAREFLRA-  ...  .:... : ... ... ... ... ... ... ...	159
lysC	103 ALATS---PALTDELVSHGELMSTLLFVEILRERDVQAQWFDRVKVMRTN	149
metL	160 ERAAQPQVDEGLSYPLLQQQLLVQHPGKRLVVT-GFISRNNAGETVLLGRN .:: ... ... ... ... ... ... ... : ... ... ... ...	208
lysC	150 DRFGRAEPDIAALALAAQLQLPRLNEGLVITQGFIGSENKGRTTTLGRG	199
metL	209 GSDYSATQIGALAGVSRVTIWSDVAGVYSADPRKVKDACLPLRLDEAS     : ...:... ... ... ... ... ... ... ... ...	258
lysC	200 GSDYTAALLAEALHASRVDIWTDPVGIYTTDPRVVAKRIDEIAFAEAA	249
metL	259 ELARLAAPVLHARTLQPVSGSEIDLQLRCSYTPDQGSTRI-----E .:: ... ... ... ... ... ... ... ...	299
lysC	250 EMATFGAKVLHPATLLPAVRSDIPFVGSSKDPFRAGGTLCNKTNPPLF	299
metL	300 RVLASGTGARIVTSHDDVCLIEFQVFASQDFKLAHKIEDQILKRAQRPL .   ...: ..  ... ... :       : ..	349
lysC	300 RALALRRNQTLTLH-----SLNMLHSRGF-LA--EVFGILAR-----	334

Figure 8: Alignement Multiple de Séquences

La PAM 250 est la plus utilisé et donne la probabilité que 250 mutations soit acceptées pour 100 acides aminés.

	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	4															
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	1	-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	-4	-4	-4	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	-1	-1	-3	0	-1	-1	0	-2	-3	-1	-2	-5	6					
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	3				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-2	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	

Figure 9: Matrice de Substitution PAM 250 Utilisée pour l'Alignement de Séquences Protéiques

### 2.3.2. BLOCKS OF AMINO ACID SUBSTITUTION MATRIX

Les séquences sont regroupées dans une matrice si leur identité dépasse un certain seuil. Dans une matrice BLOSUM62, les séquences présentant plus de 62% d'identité ont été regroupées ensemble.

Quand on utilise les matrices BLOSUM pour aligner des séquences, on devrait systématiquement choisir la matrice la plus adéquate, en fonction du pourcentage de similarité.

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

Figure 10: BLOCKS OF AMINO ACID SUBSTITUTION MATRIX

Les valeurs positives indiquent des substitutions fréquentes ("acceptées"), càd des substitutions observées plus fréquemment que ce à quoi l'on s'attendrait par hasard. Les scores élevés correspondent souvent à des acides aminés partageant des propriétés physicochimiques.

Les valeurs négatives indiquent les mutations rares, càd celles qu'on observe moins fréquemment que ce à quoi l'on s'attendrait par hasard. Ce taux inférieur est interprété comme un indice de la contre-sélection, suggérant que ces mutations sont généralement défavorables pour la fonction de la protéine.

La diagonale correspond aux identités. Il constituée de scores positifs, qui reflètent le taux de conservation des résidus.

### 3. Alignement par paire de séquences

L'alignement par paires est une méthode en bioinformatique utilisée pour aligner deux séquences de nucléotides (ADN ou ARN) ou d'acides aminés (protéines) afin de maximiser leur similitude. Cet alignement permet de repérer les régions homologues entre les séquences, ce qui peut donner des indices sur leur structure, leur fonction et leur évolution.

#### 3.1. L'algorithme de Needleman et Wunsch

Utilise une approche de programmation dynamique pour remplir une matrice de scores basée sur les correspondances, les mismatches et les gaps (insertions ou délétions).

La trace de cette matrice permet de retrouver l'alignement optimal.

Dans le cas où l'utilisateur ne souhaite pas utiliser une des matrices de scores décrites précédemment (PAM, BLOSUM ...etc.), il est primordial de définir au préalable 3 scores (identité, substitution et gap) afin d'appliquer convenablement l'algorithme de Needleman et Wunsch.

$$S(i,j) = \text{Max} \left\{ \begin{array}{l} S(i-1,j-1) + se(i,j) \\ S(i-1,j) + \text{gap} \\ S(i, j-1) + \text{gap} \end{array} \right.$$

#### 3.2. Exemple d'alignement

On considère les deux séquences suivantes :

S1 = MPRCLCQR

S2 = PYRCKCR

Afin d'aligner ces deux séquences, il est important de définir le score d'identité, de substitution et de gap.

Dans cet exemple, on admet que le score d'identité = 3, le score de substitution = -1 et le score de gap = -2.

##### 1ere étape : Construction de la matrice initiale

Dans un premier temps, les deux séquences S1 et S2 sont insérées dans une matrice dite initiale de sorte que S1 soit à l'horizontal (axe des abscisses) et S2 à la verticale du tableau (axe des ordonnées). Puis, les cases de cette matrice doivent être remplies par des scores élémentaires appropriés (3 si les deux acides aminés des deux séquences sont identiques et -1 pour une substitution).

	M	P	R	C	L	C	Q	R
P	-1	3	-1	-1	-1	-1	-1	-1
Y	-1	-1	-1	-1	-1	-1	-1	-1
R	-1	-1	3	-1	-1	-1	-1	3
C	-1	-1	-1	3	-1	3	-1	-1
K	-1	-1	-1	-1	-1	-1	-1	-1
C	-1	-1	-1	3	-1	3	-1	-1
R	-1	-1	3	-1	-1	-1	-1	3

Figure 11: Construction de la matrice initiale

## 2ème étape : construction de la matrice transformée

Dans un deuxième temps, il faut créer une deuxième matrice à  $i+2$  colonnes et  $j+2$  lignes, dans laquelle la 1ère ligne et la 1ère colonne seront initialisées non pas à zéro mais en utilisant le score des gaps (-2) comme suit :

		j	1	2	3	4	5	6	7	8
i	0	-2	-4	-6	-8	-10	-12	-14	-16	
1	P	-2								
2	Y	-4								
3	R	-6								
4	C	-8								
5	K	-10								
6	C	-12								
7	R	-14								

Figure 12: Construction de la matrice transformée

C'est à ce niveau qu'on applique l'algorithme de Needleman-Wunsch afin d'aligner les deux séquences S1 et S2 :

Si on commence par la case (1,1), l'algorithme est appliqué comme suit :

$$S(1,1) = \text{Max} \left\{ \begin{array}{l} S(0,0) + se(1,1) = 0 + -1 = -1 \\ S(0,1) + -2 = -4 \\ S(1,0) + -2 = -4 \end{array} \right.$$

Le maximum entre -1, -4 et -4 est bien -1. Donc le score à mettre dans la case i= 1 et j=1 c'est -1.

		M	P	R	C	L	C	Q	R
	0	-2	-4	-6	-8	-10	-12	-14	-16
P	-2	-1							
Y	-4								
R	-6								
C	-8								
K	-10								
C	-12								
R	-14								

Figure 13: L'application de l'algorithme de Needleman-Wunsch

L'application de l'algorithme de Needleman-Wunsch permet de remplir la matrice transformée comme suit :

		M	P	R	C	L	C	Q	R
	0	-2	-4	-6	-8	-10	-12	-14	-16
P	-2	-1	1	-1	-3	-5	-7	-9	-11
Y	-4	-3	-1	0	-2	-4	-6	-8	-10
R	-6	-5	-3	2	0	-2	-4	-6	-5
C	-8	-7	-5	0	5	3	1	-1	-3
K	-10	-9	-7	-2	3	4	2	0	-2
C	-12	-11	-9	-4	1	2	7	5	3
R	-14	-13	-11	-6	-1	0	5	6	8

Figure 14: Résultat de l'algorithme de Needleman-Wunsch

### 3ème étape : traceback (traçage en arrière)

le parcours de la matrice transformée commence par le plus haut score, vers le plus haut score parmi les trois cases  $(i-1, j-1)$   $(i-1, j)$  et  $(i, j-1)$  et ainsi de suite jusqu'à la case  $(1,1)$ . Dans cet exemple, on commence par la case  $(7,8)$  ayant le plus haut score = 8. Dans ce cas, les scores des 3 cases  $(6,7)$   $(6,8)$  et  $(7,7)$  sont respectivement 5, 3 et 6. Donc, le parcours de la matrice sera vers la case  $(7,7)$  où le score est le plus grand soit 6. Le parcours final de la matrice transformée est le suivant :

	S1	M	P	R	C	L	C	Q	R
S2	0	-2	-4	-6	-8	-10	-12	-14	-16
P	-2	-1	1	-1	-3	-5	-7	-9	-11
Y	-4	-3	-1	0	-2	-4	-6	-8	-10
R	-6	-5	-3	2	0	-2	-4	-6	-5
C	-8	-7	-5	0	5	3	1	-1	-3
K	-10	-9	-7	-2	3	4	2	0	-2
C	-12	-11	-9	-4	1	2	7	5	3
R	-14	-13	-11	-6	-1	0	5	6	8

Figure 15: Traceback dans une matrice de scores pour l'alignement de séquences

#### 4ème étape : génération de l'alignement et calcul des scores

En suivant le parcours de la matrice transformée tracé précédemment, les acides aminés en diagonal représentent soit appariement (identité) ou une substitution ( : ).

Il est très important de signaler que les acides aminés en horizontal dans le parcours de la matrice transformée représentent soit une insertion dans la séquence S1 ou une délétion dans la séquence S2. Par exemple, le premier et le deuxième acides aminés de la séquence S1 (M et P respectivement), sont en horizontal dans le parcours de la matrice transformée. Cela signifie que soit la séquence S1 a subit une insertion de l'acide aminé M (car M a un plus faible score par rapport à P) ou alors la séquence S2 a subit une délétion de cette acide aminé au cours de l'évolution.

De même, les acides aminés en vertical représentent soit une insertion dans la séquence S2 ou une délétion dans la séquence S1. Par exemple, le deuxième et le troisième acide aminé de la séquence S2 (Y et R respectivement) sont en vertical dans le parcours de la matrice transformée. Cela signifie que soit il y'a eu une insertion de l'acide aminé Y (plus faible score par rapport à l'acide aminé R) dans la séquence S2, ou alors une délétion de cette acide aminé dans la séquence S1 et ce par nécessité adaptative.

S1	M	P	-	R	C	L	C	Q	R
S2	*		*			:		*	
S2	-	P	Y	R	C	K	C	-	R

Figure 16: Génération de l'alignement

Le trou ( \_ ) en position 3 entre l'acide aminé P et R de la séquence S1 représente un gap ou indel. Il est représenté en \* lors de l'alignement.

Il est également important de signaler qu'une substitution a eu lieu au niveau de la sixième position où l'acide aminé L de la séquence S1 a été remplacé par un K dans la séquence S2 et ce par nécessité évolutive et d'adaptation à l'environnement.

### 3.3. Calcul des scores

**Le pourcentage d'identité (%id)** = (nombre d'identités / taille de la séquence après alignement)

Dans cet exemple %id= (5/9)\* 100 = 55.55%

**Le pourcentage des gaps** := (nombre gaps / taille de la séquence après alignement)\*100  
= (3/9)\*100= 33.33%

**Le pourcentage des substitutions** : (nombre de substitutions / taille de la séquence après alignement)\*100  
= (1/9)\*100= 11.11%.

**Score d'alignement** : (nombre d'identités \* score d'identité) + (nombre de substitutions\*score des substitutions) + (nombre de gaps \* score des gaps)  
(5\*3)+(1\*-1)+(3\*-2)=8

### **3.4. Utilisation de l'Alignement par Paires de Séquences**

L'alignement par paires de séquences est une technique fondamentale en bioinformatique largement utilisée dans de nombreux domaines de la recherche en biologie moléculaire. Voici quelques cas d'application clés où cette méthode est indispensable :

#### **3.4.1. Comparaison de Gènes**

L'alignement par paires de séquences est essentiel pour comparer des gènes entre différentes espèces ou individus d'une même espèce. En alignant les séquences génétiques, les biologistes peuvent identifier les régions conservées et les régions variables, ce qui permet de comprendre l'évolution des gènes et des organismes. Cette analyse comparative est cruciale pour étudier les relations évolutives entre les espèces, détecter les homologies fonctionnelles et prédire la fonction des gènes inconnus en se basant sur la conservation des séquences.

#### **3.4.2. Étude de Mutations et de Variantes Génétiques**

L'alignement par paires de séquences est également utilisé pour étudier les mutations et les variations génétiques. En alignant les séquences d'ADN ou d'ARN de différentes sources, les chercheurs peuvent identifier les variations ponctuelles telles que les substitutions, les insertions et les délétions. Cela permet de détecter les mutations responsables de maladies génétiques, d'étudier la diversité génétique au sein d'une population et de suivre l'évolution des virus et des agents pathogènes.

#### **3.4.3. Motivation de l'Utilisation de l'Alignement par Paires**

L'alignement par paires de séquences offre plusieurs avantages cruciaux dans ces applications:

- **Identification de l'Homologie** : En alignant les séquences, les similitudes entre les séquences homologues peuvent être mises en évidence, permettant ainsi d'identifier les relations évolutives et les fonctions conservées.
- **Détection de Motifs Conservés** : Les régions conservées entre les séquences alignées peuvent révéler des motifs fonctionnels importants, tels que les sites de liaison à des protéines ou à des régions régulatrices.
- **Analyse de la Diversité Génétique** : L'alignement par paires permet de détecter les variations génétiques et de quantifier la diversité génétique au sein d'une population, ce qui est crucial pour les études évolutives et de génétique des populations.
- **Prédiction de la Fonction** : En identifiant les régions conservées entre les séquences, l'alignement par paires peut aider à prédire la fonction des gènes et des protéines inconnus en se basant sur la conservation des séquences.

### 3.5. Implémentation

Le script Python permet d'aligner deux séquences d'acides aminés en utilisant la matrice de substitution BLOSUM62. Le programme commence par demander à l'utilisateur d'entrer la taille des deux séquences, puis génère ces séquences de manière aléatoire en utilisant une liste de 20 acides aminés standards. Une matrice est ensuite initialisée avec des dimensions basées sur les longueurs des deux séquences. Les scores de correspondance, de non-correspondance et de trou sont définis, et la première ligne ainsi que la première colonne de la matrice sont remplies avec des pénalités de trou. Le script remplit ensuite la matrice de scores en utilisant les valeurs de BLOSUM62 pour calculer les scores de correspondance entre les acides aminés des deux séquences. Une fois la matrice remplie, le programme procède à un backtracking pour déterminer l'alignement optimal des deux séquences. Les séquences alignées sont générées en suivant le chemin de scores maximaux dans la matrice. Le score d'alignement est ensuite calculé en additionnant les scores des paires d'acides aminés alignés. Enfin, le script affiche les séquences alignées, le score d'alignement et le temps d'exécution. Vous trouverez le code source complet dans l'annexe du document.

### 3.6. Résultat de l'alignement

Après avoir entré les tailles des deux séquences (10 chacune), les séquences générées sont : "CWWTTFHEGK" pour la première et "YQLNDFSWIY" pour la deuxième. Les séquences alignées sont obtenues par un processus de backtracking à partir d'une matrice de scores remplie en utilisant la matrice de substitution BLOSUM62. Le résultat de l'alignement montre que la première séquence alignée est "CW-WTTFHEGK-" et la deuxième séquence alignée est "-YQLNDF-SWIY". Le score de cet alignement est -2, et le temps d'exécution de l'algorithme est de 0.0 secondes.

```
Entrez la taille de la première séquence : 10
Entrez la taille de la deuxième séquence : 10
Séquence 1 : CWWTTFHEGK
Séquence 2 : YQLNDFSWIY
Séquence 1 alignée: CW-WTTFHEGK-
Séquence 2 alignée: -YQLNDF-SWIY
Score d'alignement: -2
Temps d'exécution: 0.0 secondes
```

Figure 17: Résultat de l'exécution du programme d'alignement

### 3.7. Tests sur des séquences de différentes longueurs

Le tableau récapitulatif présente les résultats de l'alignement de séquences, avec les colonnes indiquant la taille de la première séquence, la taille de la deuxième séquence, la taille globale, le temps d'exécution et le score d'alignement.

Sequence 1	Sequence 2	Taille Globale	Temps d'execution	Score d'alignement
10	10	20	0.001006	4
10	20	30	0.001045	7
20	20	40	0.000000	12
30	30	60	0.000996	21
50	50	100	0.001004	33
100	100	200	0.001004	91
200	200	400	0.002000	230
500	500	1000	0.003999	546
1000	1000	2000	0.009006	1196

Nous présentons une partie des résultats d'exécution comprenant l'alignement effectué, le score obtenu et le résultat de cet alignement.

```

    Entrer pour continuer, n pour quitter) : o
Entrez la taille de la première séquence : 20
Entrez la taille de la deuxième séquence : 20
Séquence 1 : VFNQLSMYNGETQKAAMTTE
Séquence 2 : SHDFHMNLLIIDDCCSDKDFTK
Séquence 1 alignée: --VF--NQLSMYNGETQKAAMTTE
Séquence 2 alignée: SHDFHMN-LLIIDDCCS-DK--DFTK
Score d'alignement: 12
Temps d'exécution: 0.0 secondes
Voulez-vous faire un alignement? (o pour continuer, n pour quitter) : o
Entrez la taille de la première séquence : 30
Entrez la taille de la deuxième séquence : 30
Séquence 1 : IRHTRMYAMFEMGYYSESWWAHTDVMVDT
Séquence 2 : IHFWPMSCGQFLGMKKTGFTCTIMNRVGAP
Séquence 1 alignée: IRHT-RM-YAMFEMGYYSESWWAHTDVM--V-DT
Séquence 2 alignée: I-HFWPMSCGQF-LG-MKKT-GFTCT-IMNRVGAP
Score d'alignement: 21
Temps d'exécution: 0.0009963512420654297 secondes
Voulez-vous faire un alignement? (o pour continuer, n pour quitter) : o
Entrez la taille de la première séquence : 50
Entrez la taille de la deuxième séquence : 50
Séquence 1 : PNTFFHDCPHFISCYNGTENNSIRREWIRMMPGVTKYTHNCHIFCPFIL
Séquence 2 : YAMRQMAPPLAMPMPFSPWPVCVQYKCTVPWLVMQYDTRDAGAAVGLN
Séquence 1 alignée: -----P-N--TF-FHDCPHFISCYNGTENNSIRREWIRMMPGVTKY-THNCHIFCPFIL
Séquence 2 alignée: YAMRQMAPPLAMPMPFSPWP-CV-CYQYKCT---V--PWL-VM---QYDTRDAGAAV-GLN
Score d'alignement: 33
Temps d'exécution: 0.0010035037994384766 secondes

```

Figure 18: Extrait des résultats d'exécution de l'alignement

### 3.8. Analyse de la complexité expérimentale

#### 3.8.1. Temps d'exécution constant pour petites séquences

Pour des tailles de séquences allant jusqu'à 100 nucléotides (taille globale de 200), le temps d'exécution reste presque constant, autour de 0,001 secondes. Cela suggère que l'algorithme est optimisé pour des petites séquences ou que les opérations requises sont assez légères pour ces tailles.

#### 3.8.2. Augmentation du temps d'exécution avec la taille des séquences

Au-delà de 100 nucléotides, le temps d'exécution commence à augmenter de manière plus significative. Par exemple :

- Pour des séquences de taille 200 (taille globale 400), le temps d'exécution double à 0,002 secondes.
- Pour des séquences de taille 500 (taille globale 1000), le temps quadruple à 0,004 secondes.
- Pour des séquences de taille 1000 (taille globale 2000), le temps augmente encore plus, atteignant 0,009 secondes.

#### 3.8.3. Comportement linéaire à quadratique

L'augmentation du temps d'exécution semble plus proche d'une relation quadratique pour des séquences de taille moyenne à grande. Cela peut être dû à la nature de l'algorithme d'alignement, qui typiquement présente une complexité temporelle de  $O(n^2)$  à  $O(n^3)$  pour les algorithmes de type Needleman-Wunsch ou Smith-Waterman, couramment utilisés pour les alignements globaux et locaux.

#### 3.8.4. Variabilité et précision des mesures

Un temps d'exécution de 0 secondes pour des séquences de taille 20 peut être une anomalie de mesure ou une indication que certaines optimisations (comme les opérations en mémoire cache) ont rendu l'exécution extrêmement rapide pour cette taille spécifique.

## 4. Alignement multiple de séquences

L'alignement multiple permet de détecter les régions qui ont été conservées au travers de l'évolution. Très souvent ces régions correspondent à des domaines associés à une fonction clé de la molécule.

Les AA strictement conservés, comme ceux qui apparaissent en vert, jouent souvent un rôle direct dans sa fonction.

L'alignement multiple est effectué à l'une des utilisations suivantes :

- Identifier les régions de forte similarité et de différence : Les régions conservées peuvent avoir des fonctionnalités importantes.
- Créer une séquence consensus, Contribuer à la prédiction des structures secondaires et tertiaires de nouvelles séquences.
- L'alignement multiple est une étape préliminaire pour la construction d'arbres phylogénétiques (dendrogrammes).

## Multiple sequences Alignment (MSA)

1KWP.pdb	-----FAKEETTS-----	GPEKYD-----	KSCDMWSLGIVIMYILLCGYPPFYSNH	203
1NXK.pdb	-----FAKEETTPYYVAAPEVLGPEKYD-----	KSCD-WSLG-VIYILLCGYPPFYSNH	207	
1AO6.pdb	-----VAPEVLA-----	KAVDCWSIGVIAVILLCGYPPFYDEN	185	
1KOA.pdb	LDPKQSVKVITGTAEFAAPEVAEGKPVG-----	YYTDMWSVGVLSYILLSGLSPPFGGEN	229	
1KOB.pdb	LNPDIEIVKVITTAEEFAAPEIVDPEPVG-----	FYTDMWAIAGVVLGVVLLSGLSPFAGED	232	
1JKK.pdb	IDFGNEFKNIFGTPFVAAPEIVVNTPEPLG-----	LEADMUSIGVITYILLSGASPPFLGDT	220	
1QL6.pdb	LDPGEKLRSVCGTPSYLAPEIIECSMNDNHPGYGKEVDWSTGVIMYVTLLAGSPPFWHRK	223		
1F3M.pdb	-----ITTMVGTYPWMAPEVVTRKAYG-----	PKVDIWSLGIMAIEMIEGEPPVLINE	283	
1CMK.pdb	VKGRTWT-LCGTPEYLAPEIILS-KGYNK-----	AVDUWALGVLIYEMAAGYPPFFAD	241	
1YDT.pdb	VKGRTW-LCCTPEYLAPEIILS-KGYNK-----	AVDUWALGVLIYEMAAGYPPFFAD	226	
1Q61.pdb	VKGRTW-LCCTPEYLAPEIILS-KGYNK-----	AVDUWALGVLIYEMAAGYPPFFAD	227	
2CPK.pdb	VKGRTWT-LCCTPEYLAPEIILS-KGYNK-----	AVDUWALGVLIYEMAAGYPPFFAD	227	
1FOT.pdb	VDPVTY--LCCTPDYIAPEVVST-KPYNK-----	SIDWWSFGILLYEMLAGYTPFYDS	203	
super_STK_1.pdb	VDPVTY--LCCTPDYIAPEVVST-KPYNK-----	SIDWWSFGILLYEMLAGYTPFYDS	480	
1U99.pdb	LN-----FVGTAQYVSPELLTE-KSACK-----	SSDWLWALGCIIYQLVAGLPPFRAG	204	
1GZK.pdb	-----TPEYLAPEVLED-NDYGR-----	AVDUWGLGVVMYEMMCGRRLFYNN	182	
1OMW.pdb	FSKCKPHASVGTGYMAPEVLQKGVAYDS-----	SADUFSLGCMLFKLLRGHSPPRQHK	367	
1MQ4.pdb	APSSRT--LCGTLDYLPPEMIEG-PMHDE-----	KVDLWUSLGVLCYEFLVGKPFPEAN	205	
1LUF.pdb	IYSA--DYYVDAPIR--WMPPEIIFYN-----	RYTTESDVUAYGVVLUEIFS-----	211	
super_STK.pdb	IYSA--DYYVDAPIR--WMPPEIIFYN-----	RYTTESDVUAYGVVLUEIFS-----	7453	
1O9U.pdb	LVRG--EPNVS-ICSRY-YRAPELIFGAT-----	DYTSSIDWVSAGCVLAELLLGQPIFFGDS	226	
1PYX.pdb	LVRG--EPNVSYICSRY-YRAPELIFGAT-----	DYTSSIDWVSAGCVLAELLLGQPIFFGDS	222	
1IA8.pdb	FRYMMNRERLLNMCGTLFPVVAPELLKRRE--	FHAEPVDVUSCGIVLTAMLAG-----EL	202	
1A9U.pdb	TDEDEMTG-----YVATRUYRAPEIMLNW-MHYMQTVDIWSVGCIMAELLTGR-TLFPGT	223		
1CM8.pdb	ADSEMG-----VVTRUYRAPEVILNU-MRYTQTVDIWSVGCIMAEMITGK-TLFKG	214		
1JNK.pdb	-SFMMTP-----YVYVTRYYRAPEVILG-MGYKENVDIWSVGCIMGEMVRHK-ILFPGR	217		
1ERK.pdb	ADPDHDHTGFLTEYVATRUYRAPEIMLNS-KGYTKSIDIWSVGCILAEMLSNR-PIFPKG	228		
1PME.pdb	ADPDHDHTGFLTEYVATRUYRAPEIMLNS-KGYTKSIDIWSVGCILAEMLSNR-PIFPKG	212		
1GII.pdb	F-----VTLWYRAPEILLGC-KYYSTAVD IWSLGCIFAEMVTRR-ALFPGD	187		
1PF8.pdb	FGVPVRTYTH---EVVTLWYRAPEILLGC-KYYSTAVD IWSLGCIFAEMVTRR-ALFPGD	206		
1H4L.pdb	FGIPVRCYSA---EVVTLWYRPPDVLFKA-KLYSTSIDMWSAGCIFAELANAARPLFPGN	197		
1VOB.pdb	FGIPVRCYTH---EVVTLWYRAPDVLMGS-KKYSTTID IWSVGCIFAEMVNGA-PLFPGV	204		
1B18.pdb	YSFQMALTS---VVVTLWYRAPEVLLQS-S-YATPVDLUSVGCIFAEMFRRK-PLFRGS	185		
1DS5.pdb	YHPGKEYNVR---VASRYFKGPELLVLDL-QDYDYSLDMWISLGCMFAGHIFRKEPFFYGH	1215		
1NA7.pdb	YHPGQEYNVR---VASRYFKGPELLVDY-QMYDYSLDMWISLGCMMLASMIFRKEPFFHGH	233		

Figure 19: Exemple d'un alignement multiple de séquences

#### 4.1. Exemple d'alignement multiple

Le score d'un alignement multiple est une mesure de la qualité de cet alignement, basée sur différents facteurs tels que la similarité des séquences alignées, la présence de gaps (trous) dans l'alignement, etc.

Habituellement, lorsqu'on évalue la qualité d'un alignement multiple, on considère que les colonnes dans l'alignement sont indépendantes les unes des autres. Cela signifie que chaque colonne de l'alignement représente une position dans les séquences alignées, et que l'insertion d'un gap dans une colonne n'affecte pas les scores des autres colonnes. Cette hypothèse simplifie l'évaluation de la qualité de l'alignement en considérant chaque position de manière isolée.

$$s(x, x) = 1, s(x, y) = -1, s(x, -) = s(-, x) = -2, s(-, -) = 0$$

A	A	C	G	T	A	C	G	A	T	A	
A	-	C	G	T	A	-	A	A	T	G	
G	T	C	G	T	A	-	-	T	T	A	
<hr/>											
(1-2)	1	-2	1	1	1	-2	-1	1	1	-1	
(1-3)	-1	-1	1	1	1	-2	-1	-1	1	1	
(2-3)	-1	-2	1	1	1	0	-2	-1	1	-1	
	=	=	=	=	=	=	=	=	=	=	
	-1	-5	3	3	3	-4	-5	-1	3	-1	= -2

Figure 20: Calcul d'un score d'alignement multiple

#### 4.2. Utilisation de l'alignement multiple

L'alignement multiple de séquences est une technique essentielle dans de nombreux domaines de la bioinformatique et de la biologie moléculaire en raison de ses nombreuses applications pratiques. Voici quelques cas d'application qui illustrent l'importance et la pertinence de l'alignement multiple :

- **Identification de gènes et d'homologies fonctionnelles** : L'alignement multiple permet de comparer les séquences génétiques de différentes espèces pour identifier les gènes homologues. Cette comparaison permet de déterminer les similitudes et les différences entre les séquences, ce qui peut aider à prédire la fonction des gènes inconnus en se basant sur la conservation des régions fonctionnelles.

- **Étude de l'évolution moléculaire** : En alignant les séquences de différentes espèces ou de différents individus d'une même espèce, les biologistes peuvent étudier les changements évolutifs qui se sont produits au fil du temps. L'alignement multiple permet d'identifier les sites conservés et les sites variables, ce qui aide à reconstruire l'histoire évolutive des organismes.
- **Conception de médicaments et de vaccins** : L'alignement multiple est utilisé pour identifier les régions conservées des protéines pathogènes, telles que les virus ou les bactéries, afin de concevoir des médicaments ou des vaccins ciblant ces régions. En identifiant les parties communes aux différentes souches d'un pathogène, les scientifiques peuvent développer des traitements efficaces contre une gamme de variants.
- **Étude des mutations et des maladies génétiques** : L'alignement multiple est utilisé pour comparer les séquences génétiques de patients atteints de maladies génétiques avec celles de personnes en bonne santé. Cela permet d'identifier les mutations responsables des maladies et de comprendre leur impact sur la structure et la fonction des protéines.
- **Annoter les génomes et prédire les fonctions des gènes** : L'alignement multiple est utilisé pour annoter les génomes en identifiant les régions codantes, les régions régulatrices et les éléments structuraux importants. En comparant les séquences génomiques avec des bases de données de séquences connues, les chercheurs peuvent prédire les fonctions des gènes et des éléments non codants.

### 4.3. Méthodes d'alignement multiple

Dans la littérature, on rencontre trois catégories essentielles ou approches suivies pour construire un MSA. Néanmoins, ces approches sont parfois fusionnées, concaténées ou/et associées pour construire une seule méthode.

On distingue l'approche Exacte qui tente de donner plus de longévité à la programmation dynamique dans ce domaine et de déterminer un alignement optimal proprement dit comme elle le fait pour aligner deux séquences. De l'autre côté, on rencontre des heuristiques qui à leur tour se bifurquent en deux approches : Progressive et Itérative.

#### 4.3.1. L'Approche Exacte

Les algorithmes d'alignement multiples de séquences exacts permettent de réaliser des alignements d'un petit nombre de séquences. L'approche exacte n'est autre qu'une généralisation des méthodes de programmation dynamique de Needleman et Wunsh, et Smith et Waterman.

La méthode de programmation dynamique utilisée pour aligner deux séquences, a été appliquée à l'alignement de plusieurs séquences (N dimensions) tels que MSA et DCA.

Ce type de méthodes représente de gros problèmes : Le temps de calcul et l'espace mémoire.

Dans la pratique, un alignement devient délicat pour un nombre de séquence  $N > 3$ , et même impossible pour  $N = 10$ . - Pour  $N$  séquences de longueur  $L$ , l'alignement optimal (au sens mathématique) nécessite :

- Un temps de calcul proportionnel à  $(2^n)(L^n)$
- Un espace mémoire proportionnel à  $(L^n)$

#### 4.3.1.1. La Programmation dynamique

La programmation dynamique est un principe souvent simple à mettre en œuvre pour résoudre des problèmes complexes. Elle ne s'applique toutefois qu'à une certaine catégorie de problèmes, et il est nécessaire de vérifier certaines conditions pour qu'elle puisse être appliquée.

Soit  $P(n)$  un problème satisfaisant au principe d'optimalité. On dit qu'un algorithme de résolution de  $P(n)$  est basé sur le principe de la programmation dynamique s'il utilise les deux étapes suivantes :

- $P(n)$  est calculé récursivement en partant des problèmes de plus bas niveau.
- Une table est construite dynamiquement pour conserver tous les résultats intermédiaires obtenus.

L'utilisation de la programmation dynamique suppose donc que pour les valeurs les plus faibles de  $n$ ,  $P(n)$  soit connu ou facile à calculer. En conservant tous les résultats intermédiaires dans une table on évite d'avoir à les recalculer de nombreuses fois. En effet, refaire ainsi les mêmes calculs à chaque itération est à l'origine de la complexité exponentielle des algorithmes "naïfs".

#### 4.3.1.2. Avantages

**Précision élevée** : Cette méthode garantit un alignement optimal des séquences en termes de score.

**Convient pour de petites séquences** : Pour un petit nombre de séquences ou des séquences relativement courtes, l'approche exacte peut fournir des résultats optimaux.

#### 4.3.1.3. Limites

**Complexité computationnelle** : L'alignement exact de multiples séquences devient rapidement impraticable en raison de l'explosion combinatoire des possibilités d'alignement, rendant cette approche inefficace pour un grand nombre de séquences ou des séquences de grande taille.

**Sensibilité au bruit** : Les alignements basés sur la programmation dynamique peuvent être sensibles au bruit ou à la présence d'insertions ou de délétions dans les séquences, ce qui peut conduire à des résultats moins précis dans certains cas.

#### 4.3.2. Méthodes Itérative

L'approche itérative a été employée plusieurs fois comme méthode d'optimisation pour produire des alignements multiples. Parfois elle est utilisée seule ou en combinaison avec d'autres

méthodes. L'itération a un grand avantage parce qu'elle est souvent très simple soit en termes de code des algorithmes soit en termes de complexité temporelle et spatiale [47].

Les étapes d'un alignement itératif :

- Repérer les deux séquences avec la plus forte similarité et les aligner avec une méthode de programmation dynamique.
- Trouver la séquence qui est la plus proche du profil obtenu avec les 2 séquences précédentes et l'aligner avec les deux autres par une méthode d'alignement profil séquence.

#### 4.3.2.1. La Méthode DIALIGN

DIALIGN est une méthode pour l'alignement multiple développée par Morgenstern. L'algorithme de DIALIGN est basé sur les alignements par paires de séquence (alignement deux à deux) et multiple en comparant des segments entiers de séquences au lieu d'une traditionnelle comparaison de chaque résidu.

Des alignements par paires sont construits de pairs segments de même longueur sans insertion ou déletion de gaps. Ces paires de segments s'appellent les ‘diagonales’ ou (motif) observable sur le graphe d'un DOTPLOT. Par conséquent DIALIGN n'emploie aucune pénalité de gap.

Une fois une diagonale est considérée dans un alignement, elle est fixe et ne peut pas être enlevée à une étape postérieure de l'algorithme. Une diagonale n'est pas choisie selon son poids, mais plutôt selon si le motif décrit par cette diagonale, apparaît dans plus de deux séquences, alors il est préféré aux motifs qui apparaissent dans seulement deux séquences.

Cette approche est particulièrement efficace et convenable pour la détection d'une homologie locale. Sa consommation en termes de durée de calcul et en espace mémoire est considérée raisonnable.

Dalign-t c'est une version plus récente de Dalign-2, locale et progressive.

#### 4.3.2.2. La Méthode KALIGN

L'algorithme de Kalign suit une stratégie analogue à la méthode progressive standard d'alignement de séquence. Les distances par paire sont calculées, un arbre guide est construit et les séquences/profils sont alignés dans l'ordre donné par l'arbre. Contrairement aux méthodes existantes, l'algorithme de couplage approximatif de chaînes de Wu-Manber est utilisé dans le calcul de distance et, en option, dans la programmation dynamique utilisée pour aligner les profils.

C'est un nouvel algorithme d'alignement de séquences multiples basé sur la correspondance approximative de modèle Wu-Manber qui combine une haute qualité avec une vitesse élevée. Par rapport aux programmes existants, Kalign a réalisé des performances beaucoup plus robustes lors de l'alignement de grandes quantités de séquences ou de séquences distantes dans un repère à grande échelle d'alignements générés. En termes d'efficacité de calcul, Kalign est supérieur aux autres méthodes, et aligne facilement des centaines de séquences en quelques minutes sur un ordinateur de bureau normal. Associé au fait que Kalign donne des alignements très précis, cela fait de Kalign une méthode globale très attrayante. La haute précision de Kalign

est due à l'utilisation innovante de l'algorithme de couplage de cordes Wu-Manber approximatif. Cela permet d'estimer avec précision les distances de séquence, même dans les cas difficiles. Des distances de séquence précises génèrent des arbres guides de bonne qualité qui, à leur tour, conduisent à de bons alignements. Dans le même temps, Wu-Manber string-matching est très rapide et réduit considérablement le temps nécessaire pour l'étape d'estimation de distance qui domine le temps d'exécution de la plupart des programmes d'alignement. La stratégie décrite ici peut, en principe, être appliquée à toute autre méthode d'alignement progressif. Même si l'on ne tient pas compte des résultats de la nouvelle grande teste, les performances de Kalign sur Balibase et Prefab sont impressionnantes, surtout si l'on tient compte du fait que, contrairement à d'autres méthodes, Kalign n'a reçu aucune formation sur l'un ou l'autre des tests, et que d'autres méthodes ayant des performances similaires sont beaucoup plus lentes.

#### 4.3.2.3. Avantages

**Précision améliorée** : Les méthodes itératives peuvent converger vers des alignements optimaux en ajustant progressivement les paramètres de l'alignement.

**Adaptabilité** : Les méthodes itératives peuvent être optimisées pour des ensembles de données spécifiques en ajustant les paramètres de pénalité et les stratégies d'alignement.

#### 4.3.2.4. Limites

**Complexité algorithmique** : Certaines méthodes itératives peuvent être plus complexes et nécessiter des ressources computationnelles plus importantes, ce qui peut limiter leur applicabilité pour des ensembles de données très volumineux.

**Sensibilité aux paramètres** : Le choix des paramètres initiaux et des stratégies d'optimisation peut avoir un impact significatif sur la qualité des alignements, ce qui nécessite une expertise approfondie pour une utilisation efficace.

### 4.3.3. L'Approche Progressive

L'alignement progressif est l'heuristique la plus répandue pour aligner un grand nombre de séquences. L'alignement multiple est construit progressivement en alignant des paires de séquences suivies des paires d'alignements/profils. Un arbre guide détermine l'ordre dans lequel les séquences vont être alignées, les plus proches d'abord. Cette technique est employée dans différents packages d'alignement multiple tels que, ClustalW, et T-Coffee ...etc. Un alignement multiple progressif suit les étapes suivantes :

- a) Alignement deux à deux de toutes les séquences.
- b) Construction d'une matrice de distances entre toutes les séquences.
- c) Détermination de l'ordre selon lequel les séquences seront Alignées en notion de clustering:
  - Alignement de deux séquences.
  - Alignement d'une séquence et d'un profil.
  - Alignement de deux profils.

Exemple: on cherche à aligner un groupe de 4 séquences (déjà alignées) avec un groupe de 2 séquences (déjà alignées)

Calcul du score:	
1 PEEKSAV <b>T</b> AL	M(T, V) x w1 x w5 +
2 GEEKAAV <b>L</b> AL	M(T, I) x w1 x w6 +
3 PADKTNV <b>K</b> AA	M(L, V) x w2 x w5 +
4 AADKTNV <b>K</b> AA	M(L, I) x w2 x w6 +
5 EGEWQL <b>V</b> LHV	M(K, V) x w3 x w5 +
6 AAEKTK <b>I</b> RSA	M(K, I) x w3 x w6 +
	M(K, I) x w4 x w5 +
	M(K, I) x w4 x w6 / 8

Figure 21: Calcul d'un score d'alignement multiple - Approche progressif

Score associé à la comparaison d'un gap = 0 plus mauvais score possible

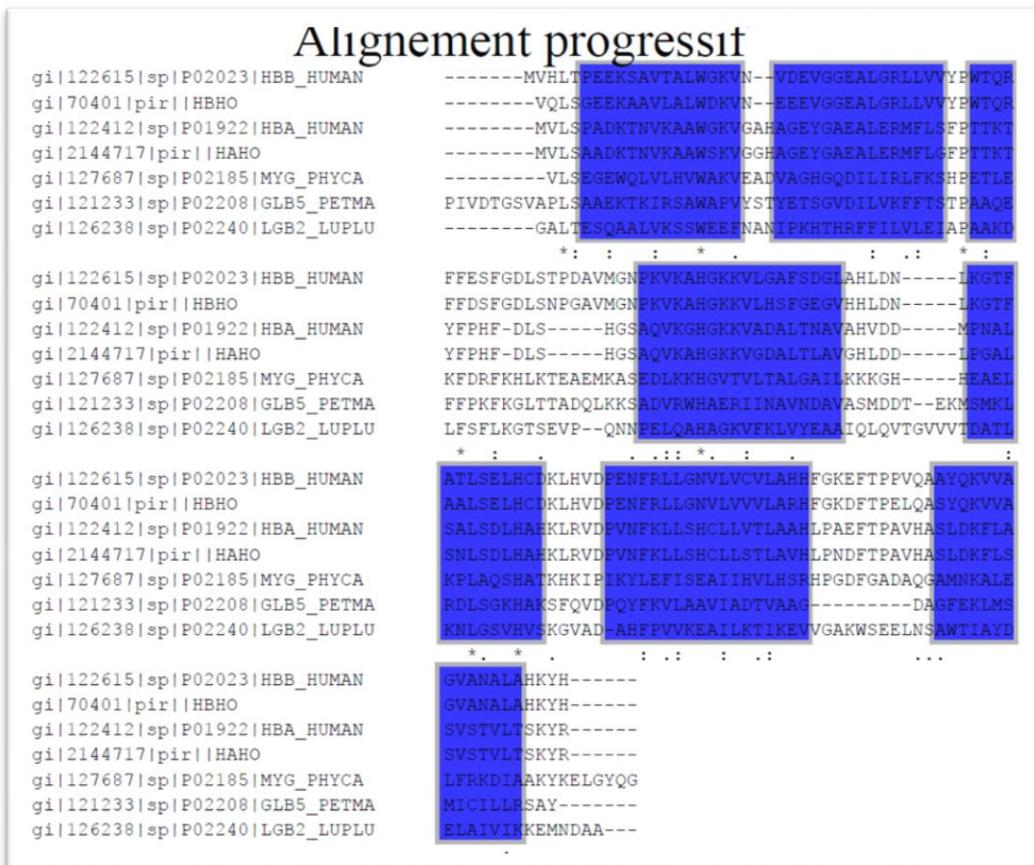


Figure 22: Alignement multiple - Approche progressif

#### 4.3.3.1. Avantages

**Évolutivité** : Cette approche est relativement efficace pour aligner un grand nombre de séquences ou des séquences de grande taille en raison de sa nature itérative et progressive.

**Flexibilité** : Les méthodes progressives peuvent être adaptées pour traiter des ensembles de données hétérogènes avec des longueurs de séquence variables.

#### 4.3.3.2. Limites

**Risque d'alignement incorrect** : Les alignements progressifs peuvent être influencés par l'ordre dans lequel les séquences sont alignées, ce qui peut conduire à des résultats suboptimaux ou même incorrects.

**Sensibilité à l'ordre d'alignement** : L'approche progressive peut être moins robuste face à des ensembles de données très disparates ou des séquences divergentes.

### 4.4. Amélioration des performances MSA dans le Big Data

Toutes les méthodes de résolution du problème MSA prennent du temps. Des recherches récentes se sont concentrées sur les moyens d'accélérer les solveurs MSA sans sacrifier la précision. Pour résoudre ce problème, diverses méthodes ont été développées, qui peuvent être divisées en deux catégories. La première est basée sur l'utilisation du parallélisme par l'approche matérielle. ClustalW-MPI, par exemple, utilise des systèmes à mémoire partagée et à mémoire distribuée, et Parallèle T-Coffee. Church et al ont fourni une conception d'algorithme MSA sur des superordinateurs avec des processeurs parallèles et une mémoire distribuée, présente une autre parallélisation de MAFFT applicable à des milliers de séquences. D'autre part, des unités de traitement graphiques ont été utilisées pour accélérer les programmes MSA, comme GPU-Blast, G-MSA, et nous pouvons citer les travaux récents de Liu et al basé sur l'alignement accéléré et l'utilisation de plusieurs GPU.

Le deuxième type de parallélisme est réalisé par une méthode logicielle qui emploie des techniques de programmation parallèle. Quatre sous-classes peuvent être proposées dans cette section :

La première est une approche parallèle dans le calcul de la matrice de notation. Cette approche a été utilisée dans Zafalon et coll où une amélioration de 15% du temps d'exécution a été observée. La deuxième est une approche Pipeline ; Agarwal et Rizvi ont proposé une technique à deux étapes pipeline qui peut améliorer la complexité du problème. Huang et coll. présentent ensuite un nouveau pipeline multi-alignement pour les données du séquençage à haut débit.

La troisième est une approche parallèle avec un algorithme dynamique. Dans la présente sous-classe, les techniques sont basées sur la parallélisation des algorithmes optimaux connus sur le terrain, par exemple, la parallélisation de l'algorithme Needleman & Wunsch par Naveed, la parallélisation de l'algorithme Smith & Waterman par Dohi et al, et la parallélisation de la technique optimale pour résoudre la MSA par Helal et al sur l'architecture GPU.

La quatrième est une technique parallèle aux données. Ce type d'étude d'optimisation est devenu de plus en plus important ces dernières années. Regrouper, répartir et aligner sont les trois étapes principales de cette technique, qui est basée sur la stratégie de division pour mieux régner. La première étape propose une approche de clustering non hiérarchique, suivie d'une répartition des clusters entre les processeurs, et enfin d'une étape d'alignement pour construire le MSA. Plusieurs travaux basés sur cette technique ont récemment été proposés, le premier travail est présenté par Fahad et al où une technique k-mer est utilisée pour faire le regroupement, mais une perte significative de la qualité de l'alignement a été observée. Plusieurs techniques de regroupement ont été proposées, comme UCLUST, CD-HIT, BlastClust et d'autres regroupements comme celui proposé dans. Ceux-ci ont permis de créer différentes approches MSA. Xiangyuan et al ont proposé une approche parallèle des données basées sur les deux systèmes de regroupement UCLUST et CD-HIT à l'étape de regroupement et MUSCLE à l'étape d'alignement.

#### 4.5. Implémentation d'une méthode progressive

##### 4.5.1. La Méthode ClustalW

ClustalW est un programme qui met en action les principes de l'alignement progressif tout en essayant d'échapper au piège des erreurs qui peuvent se produire au début de l'alignement et nuire à sa qualité dans la fin. Dans ClustalW, les auteurs essayent donc de respecter la démarche progressive mais en apportant des modifications et des nouvelles considérations.

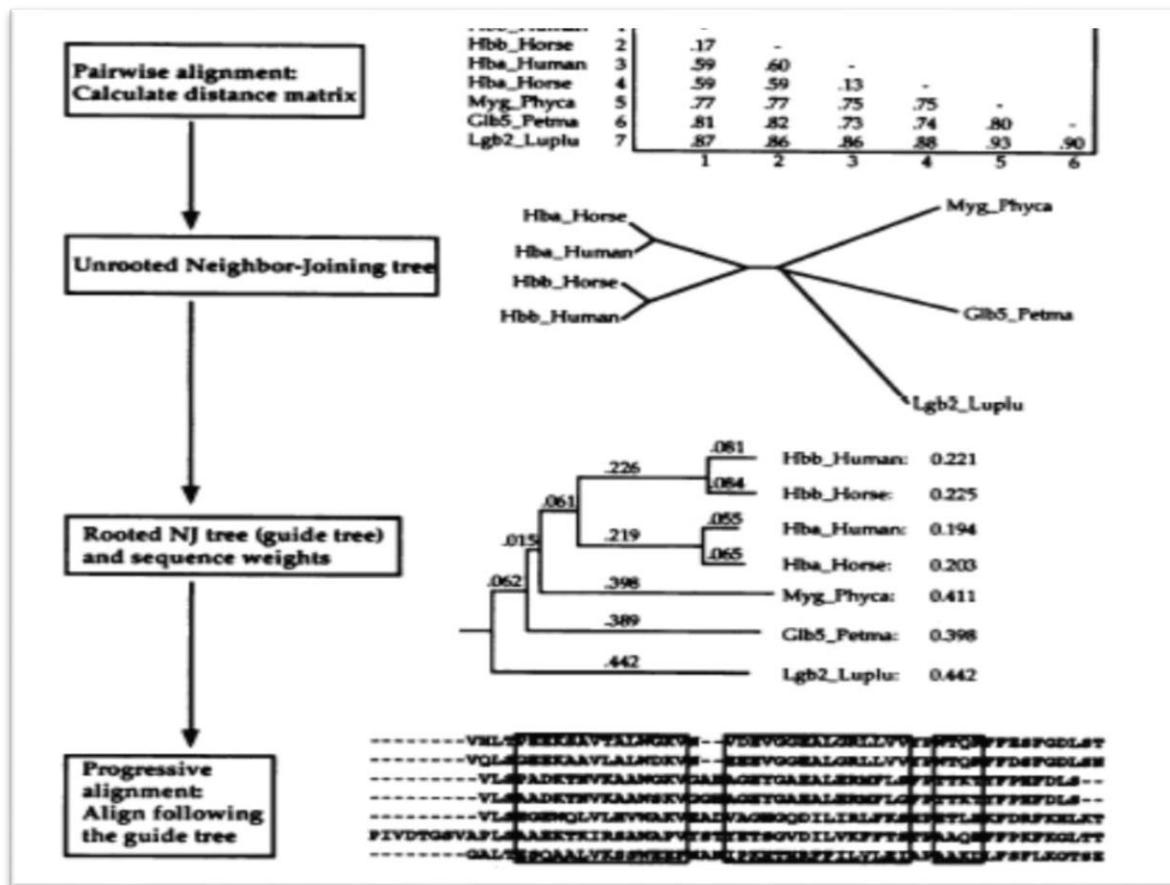


Figure 23: Processus de la méthode ClustalW

La première étape de ClustalW consiste à aligner les paires de séquences à fin de déterminer la matrice des distances. ClustalW utilise des matrices de substitutions différentes pour la programmation dynamique à des moments différents de l'alignement. Les matrices changent selon la divergence ou la convergence des deux séquences à aligner. L'avantage est que les séquences divergentes sont plus ou moins bien alignées.

Dans la deuxième étape, ClustalW utilise la méthode N.J pour construire un arbre guide et calculer les poids des séquences.

Pendant la troisième étape : alignement progressif proprement dit, ClustalW n'affecte pas la même valeur de pénalité d'un gap quel que soit sa position dans la séquence mais essayent de distinguer entre les gaps du début, du milieu et de la fin de la séquence.

Dans ClustalW, il y a une grande étude et des nouvelles propositions sur la manière de faire changer les valeurs affectées à un gap selon sa position dans une séquence ou dans un alignement de séquences.

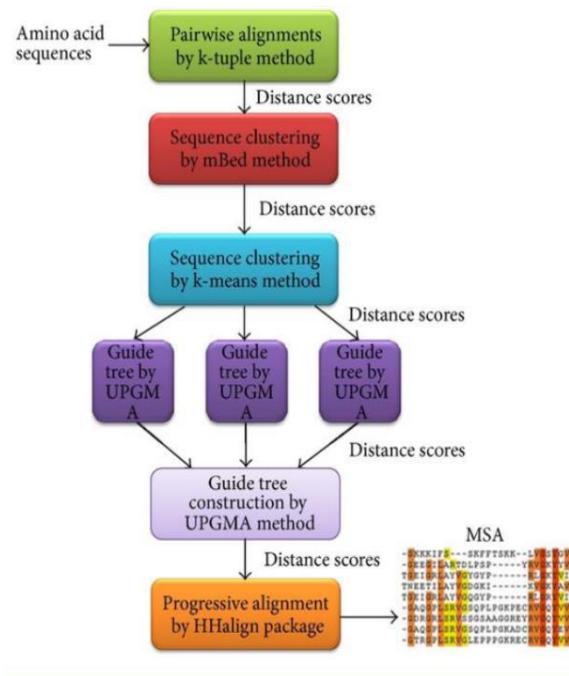
Une particularité de ClustalW est qu'il possède une interface graphique conviviale contrairement aux autres méthodes.

#### 4.5.1.1. Nouvelle version : Clustal Omega

CLUSTAL Omega utilise des arbres guides avec des graines et des techniques de profile-profile HMM pour générer les alignements.

Changement de plusieurs heuristiques (k-tuple, clustering) de séquence avec les méthodes Bed et k-means, méthode UPGMA pour la construction de l'arbre guide suivie d'un alignement progressif avec le package HHalign pour produire un l'alignement multiple.

Clustal Omega est précis et permet l'alignement d'un nombre « infini » de séquences.



*Figure 24: Processus de la méthode Clustal Omega*

#### 4.5.2. Notion de profil

Le code génère des séquences de nucléotides aléatoires et effectue un alignement multiple progressif en construisant des profils de séquence pour aligner chaque nouvelle séquence, tout en mesurant le temps d'exécution.

#### 4.5.3. Affichage les alignements obtenus progressivement

Dans cet exemple, le code a généré cinq séquences de nucléotides de 10 caractères chacune et les a alignées progressivement en construisant des profils après chaque étape. Chaque nouvelle séquence est alignée par rapport au profil obtenu jusqu'à ce point. Les séquences alignées montrent des insertions de tirets pour gérer les décalages, et les profils évoluent pour refléter les alignements accumulés. Le processus a été rapide, prenant seulement 0,01 seconde au total.

```
Entrez le nombre de séquences : 5
Entrez la taille de chaque séquence : 10
Séquences générées :
Séquence 1 : AGGTGAATT
Séquence 2 : GATAAACCT
Séquence 3 : CCCTACCTAT
Séquence 4 : GGAAAAGCGA
Séquence 5 : TGTTTGTGAC

Alignement progressif étape 1 :
Séquence 2 alignée : -----A---T
Profil 2 : AGGTGAATT

Alignement progressif étape 2 :
Séquence 3 alignée : ---T---T-T
Profil 3 : AGGTGAATT

Alignement progressif étape 3 :
Séquence 4 alignée : GGA-AAG-G-
Profil 4 : ---T-A-T-T

Alignement progressif étape 4 :
Séquence 5 alignée : TGTTT-T-A-
Profil 5 : -G-T-A-T-T
Temps total d'exécution : 0.010019 secondes
```

Figure 25: Résultat de l'exécution du programme d'alignement

## 4.6. Tests

### 4.6.1. Tests avec la longueur de séquences constante

Le tableau d'informations présente les résultats de l'exécution d'un algorithme sur plusieurs séquences de taille fixe (50) et montre comment le temps d'exécution varie en fonction du nombre de séquences alignées.

Taille séquence	Nombre de séquences	Temps d'exécution(s)
50	3	0.005997
	4	0.007995
	5	0.009996
	6	0.013001
	7	0.015674
	8	0.020004
	9	0.022006
	10	0.023264

Nous présentons ici l'un des alignements effectués :

```
Nouvel alignement :  
Entrez la taille de chaque séquence : 50  
Entrez le nombre de séquences : 5  
Séquences générées :  
Séquence 1 : TTGTAAAATTTACTCACACCGCGTCGGTAGCGATCGGTTCGACGAAGT  
Séquence 2 : AGCCTTTCTCCGCGGGCTTCCCCGTTTGCTCGTCAGCGTGATCTCGAAA  
Séquence 3 : CTGCGCTGGCACGCTAGGTGGACTGCCTGCACGGCGATCAGTACAGCT  
Séquence 4 : GATTCAATGTTCTCGAGGTAAAGGGCTCCCTACTTCGAGTCGCTAAATC  
Séquence 5 : GAGTACCTAGTCTGAGGAAACATTCTATAGTTGAGGAATCACATCTAGC  
  
Alignement progressif étape 1 :  
Séquence 2 alignée : -----T---C---C-C-----CG---G---T-----A--  
Profil 2 : TTGTAAAATTTACTCACACCGCGTCGGTAGCGATCGGTTCGACGAAGT  
  
Alignement progressif étape 2 :  
Séquence 3 alignée : -TG-----TC-----G---G---T-----G-T--G---A--T  
Profil 3 : TTGTAAAATTTACTCACACCGCGTCGGTAGCGATCGGTTCGACGAAGT  
  
Alignement progressif étape 3 :  
Séquence 4 alignée : G--TCAAT-TTC---A-GT---GGG-TC-CC--CTT-GA-TCGCTAAAT-  
Profil 4 : -TG-----T---CTC-C--CGC---G--T--CG---GG-T--G---AA-T  
  
Alignement progressif étape 4 :  
Séquence 5 alignée : G-GTA--T--TC-----AA--C-TT-TATAG--G-GG-AT---ATC-AG-  
Profil 5 : -TGT-AA-TTT-CTCAC--CGCG--G--T--CG-T-GG-TTCG--AA-T  
Temps total d'exécution : 0.009996 secondes
```

Figure 26: Extrait des résultats d'exécution de l'alignement - Approche Progressif

#### 4.6.1.1. Analyse

L'algorithme montre une complexité qui augmente avec le nombre de séquences. La nature de cette augmentation est plus que linéaire, ce qui montre que le temps d'exécution peut être influencé par des interactions supplémentaires entre les séquences à aligner, comme des comparaisons pair-à-pair multiples ou des opérations combinatoires qui augmentent le temps de calcul de manière plus marquée à mesure que le nombre de séquences augmente.

#### 4.6.2. Tests avec différents longueurs de séquences

Taille séquence	Nombre de séquences	Temps d'exécution(s)
100	3	0.006001
	4	0.008819
	5	0.010009
	6	0.013003
	7	0.017540
	8	0.011006
	9	0.023532
	10	0.027161
	3	0.005999
	4	0.009017
150	5	0.011643
	6	0.016126
	7	0.018538
	8	0.014998
	9	0.020193
	10	0.023035
	3	0.007002
	4	0.011000
	5	0.015999
	6	0.020056
200	7	0.021024
	8	0.026688
	9	0.033043
	10	0.038176
	3	0.007563
	4	0.011316
	5	0.020013
	6	0.021534
	7	0.026227
	8	0.033032
300	9	0.062521
	10	0.040670

#### 4.6.2.1. Analyse

Globalement, le temps d'exécution augmente avec le nombre de séquences et la taille des séquences. Les anomalies observées peuvent être dues à des variations dans les conditions de test ou à des spécificités non linéaires de l'algorithme en fonction de certaines combinaisons de données. Des tests supplémentaires et une analyse approfondie sont nécessaires pour expliquer ces anomalies de manière plus précise.

Ces résultats sont importants pour évaluer la performance et l'optimisation d'algorithmes d'alignement de séquences en fonction de différents paramètres, et ils soulignent la nécessité de comprendre les conditions de test et les caractéristiques des données.

### 4.7. Conclusion

L'approche progressive d'alignement multiple a été évaluée à l'aide de données de performance comprenant différentes tailles de séquences et différents nombres de séquences. Les résultats montrent que le temps d'exécution augmente généralement avec le nombre de séquences et la longueur des séquences. Cette tendance est attendue car l'approche progressive nécessite une itération à travers les séquences pour construire un alignement itératif, ce qui peut devenir plus complexe et prendre plus de temps à mesure que le nombre de séquences ou leur longueur augmente.

Concernant les limites de cette méthode, plusieurs points peuvent être soulevés :

**Optimalité des résultats** : Bien que l'approche progressive puisse converger vers un alignement global optimal dans certains cas, elle n'est pas garantie de fournir la solution optimale dans tous les cas. L'ordre dans lequel les séquences sont alignées peut influencer le résultat final, ce qui peut potentiellement conduire à des alignements sous-optimaux.

**Nombre maximum de séquences** : Bien que le programme puisse traiter jusqu'à 5000 séquences de taille 5000, il est probable que la performance de l'approche progressive diminue considérablement avec un nombre très élevé de séquences. Au-delà d'un certain seuil, le temps d'exécution pourrait devenir prohibitif et la qualité de l'alignement pourrait être compromise.

**Exactitude et performance** : L'exactitude de l'alignement dépend de plusieurs facteurs, notamment la qualité des séquences et la stratégie d'alignement utilisée. Dans certains cas, l'approche progressive peut produire des alignements moins précis par rapport à des méthodes plus sophistiquées, en particulier pour des ensembles de données très hétérogènes ou avec des séquences très divergentes.

## 5. Autre méthode d'alignement multiple

### 5.1. La Méthode SAGA (Méthodes Itérative)

C'est un algorithme génétique itératif qui démarre par une population d'alignement, puis raffine les solutions par des opérateurs spécifiques tels que la mutation jusqu'à l'obtention d'une solution plus ou moins optimale. C'est une heuristique qui se rapproche de la solution optimale mais aucune certitude qu'elle le soit réellement.

Chaque génération est évaluée par la fonction objectif (WSP) pour déterminer quels sont les alignements les plus acceptables et aptes à passer dans la génération suivante. Ceci est appelé le phénomène de la sélection biologique « seuls les meilleurs survivent ».

Initialisation	1. Créer G0
Evaluation	2. Evaluer la population de la génération n (Gn).  3. Si la population est stabilisée, FIN.
Breeding (sélection)	4. Choisir les personnes à remplacer. 5. Evaluer la descendance attendue (OE). 6. Sélectionnez-le ou les parents à partir de Gn. 7. Sélectionner l'opérateur. 8. Générer le nouvel enfant. 9. Conserver ou jeter le nouvel enfant en Gn+1. 10. Passer à 6 jusqu'à ce que tous les enfants aient été placés avec succès en Gn+1. 11. n = n+1  12. Aller à Évaluation
Fin	13. Fin

G0, Gn et Gn+1 sont respectivement la population initiale, courante et la population de la génération future. L'algorithme commence par la génération des individus de la population G0 d'une façon aléatoire, qui vont subir immédiatement une évaluation afin de déterminer le niveau de ces solutions. Si les solutions obtenues ont atteint un seuil d'optimalité alors l'algorithme s'arrête sinon on passe à l'étape suivante et qui consiste en la génération de nouvelles solutions en faisant subir à la population courante une série d'opérations génétiques telles que la sélection, croisement et mutation. Les nouvelles solutions obtenues ne sont maintenues dans la nouvelle génération que si elles présentent un certain niveau d'efficacité. L'algorithme s'arrête après un certain nombre d'itération. La meilleure solution de la dernière population serait considérée la solution optimale de l'algorithme.

SAGA a la particularité de pouvoir optimiser n'importe quelle fonction objective. Plus tard ont utilisé SAGA pour valider une nouvelle fonction objective : Coffee.

### 5.2. Implémentation

Le code implémente l'algorithme SAGA pour aligner des séquences de nucléotides de manière progressive et évolutive. Après avoir généré des séquences aléatoires de nucléotides, il crée une

population initiale d'alignements, puis procède à des mutations et croisements pour optimiser les alignements à travers plusieurs générations. À chaque génération, les alignements sont évalués et sélectionnés en fonction de leur score, avec l'objectif de maximiser le nombre de colonnes identiques dans les alignements. Enfin, le code affiche le meilleur alignement trouvé, son score, et le temps total d'exécution.

### 5.3. Tests

Taille séquence	Nombre de séquences	Temps d'exécution(s)
50	3	0.302025
	4	0.186595
	5	0.227634
	6	0.368311
	7	0.266054
	8	0.274393
	9	0.300085
	10	0.357844
	3	0.241544
	4	0.326585
100	5	0.293261
	6	0.340454
	7	0.403734
	8	0.387194
	9	0.409754
	10	0.463689
	3	0.285552
	4	0.321794
	5	0.418824
	6	0.422699
150	7	0.442606
	8	0.494291
	9	0.509848
	10	0.548218
	3	0.330859
	4	0.406071
	5	0.424856
	6	0.498104
	7	0.509691
	8	0.617193
200	9	0.810479
	10	0.933210
	3	0.436182
	4	0.511508
	5	0.631581
	6	0.652109
	7	0.736005
	8	0.769468
	9	0.804323
	10	0.913221
300		

Résultats :		
Taille de la séquence	Nombre de séquences	Temps d'exécution
50	3	0.302025
50	4	0.186595
50	5	0.227634
50	6	0.368311
50	7	0.266054
50	8	0.274393
50	9	0.300085
50	10	0.357844

Figure 27: Tableau explicatif de l'alignement - Approche itérative (SAGA)

#### 5.4. Analyse

Le temps d'exécution tend à augmenter avec la taille des séquences et le nombre de séquences à aligner.

Les variations dans le temps d'exécution peuvent être observées en fonction des paramètres de taille de séquence et de nombre de séquences.

L'algorithme SAGA semble avoir une complexité qui augmente de manière significative avec la taille des séquences et le nombre de séquences à aligner.

#### 5.5. Conclusion

L'approche itérative de l'alignement multiple de séquences, telle que la méthode SAGA (Sequence Alignments by Genetic Algorithm), a été évaluée en fonction de différentes tailles de séquences et différents nombres de séquences. Les résultats obtenus révèlent des tendances significatives en termes de temps d'exécution, mais également des limites de cette méthode.

**Optimalité des résultats :** Bien que les méthodes itératives comme SAGA visent à converger vers des solutions de meilleure qualité au fil des itérations, elles ne garantissent pas toujours l'obtention de l'alignement optimal. La qualité de l'alignement dépend de plusieurs facteurs, notamment les paramètres de l'algorithme et la sélection des métriques d'évaluation.

**Nombre maximum de séquences :** Les performances de SAGA semblent décroître avec l'augmentation du nombre de séquences, la méthode pourrait rencontrer des difficultés avec un nombre très élevé de séquences. Au-delà d'une 2500 séquence, le temps d'exécution pourrait augmenter considérablement, ce qui pourrait rendre l'application de SAGA impraticable pour de très grands ensembles de données.

**Exactitude et performance :** L'exactitude de l'alignement dépend de la qualité de l'algorithme et des paramètres utilisés. Bien que SAGA puisse fournir des résultats acceptables dans de nombreux cas, son efficacité peut être limitée par la complexité des séquences ou la présence de régions hautement divergentes.

## 6. Comparaison entre les deux méthodes

Pour comparer la méthode SAGA (itérative) et la méthode progressive en termes d'alignement multiple de séquences, nous pouvons prendre en compte plusieurs aspects :

### 6.1. Temps d'exécution

L'analyse des temps d'exécution montre que l'approche progressive est beaucoup plus efficace en termes de rapidité que la méthode SAGA. Cette différence de performance est conséquente, la méthode SAGA prenant en moyenne entre 19 et 27 fois plus de temps pour traiter les mêmes tailles de séquences. Bien que la méthode SAGA puisse offrir d'autres avantages comme une meilleure précision ou une flexibilité dans certains cas, son temps d'exécution beaucoup plus élevé la rend moins appropriée pour des applications nécessitant un alignement rapide des séquences.

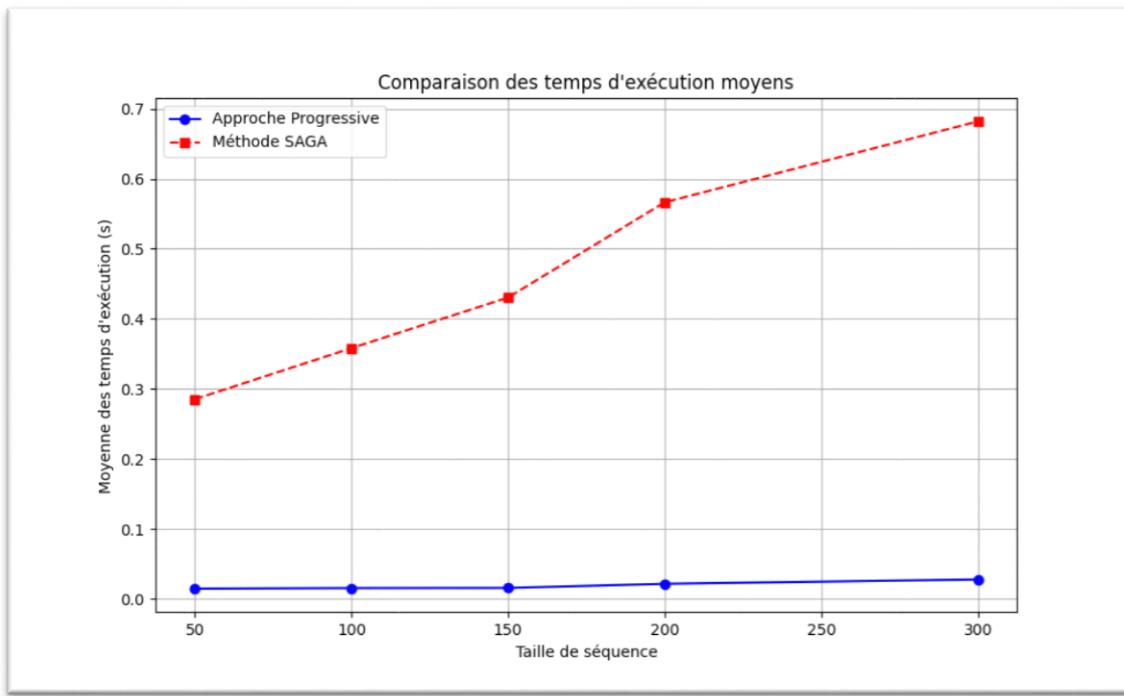


Figure 28: Comparaison entre les deux méthodes en fonction de temps d'exécution

## 6.2. Qualité de l'alignement

En comparant les résultats des alignements, il est clair que la méthode SAGA produit des alignements de meilleure qualité. Les alignements sont plus consistants, les séquences sont mieux alignées avec moins d'espaces vides, et les motifs conservés sont mieux préservés.

La méthode Approche Progressive produit des alignements moins consistants avec des séquences fragmentées et une faible préservation des motifs.

## 6.3. Scalabilité

En termes de scalabilité, l'Approche Progressive se démarque comme étant plus robuste et capable de gérer des ensembles de données beaucoup plus importants par rapport à la méthode itérative SAGA. Alors que SAGA est adaptée pour des applications avec des ensembles de données de taille modérée, l'Approche Progressive est préférable pour des analyses à grande échelle.

Méthode	Nombre maximal de séquences	Taille maximale des séquences
Approche Progressive	5000	5000
SAGA	1500	1000

## 7. Conclusion

bio-informatique, et en particulier l'alignement de séquences, joue un rôle crucial dans la compréhension de la biologie moléculaire et de l'évolution des organismes vivants. Dans ce mini-projet, nous avons exploré deux aspects essentiels de l'alignement de séquences : l'alignement par paire et l'alignement multiple.

Tout d'abord, nous avons étudié l'alignement par paire, en mettant en œuvre l'algorithme classique de Needleman et Wunsch. Nous avons montré comment cet algorithme fonctionne en alignant deux séquences, calculant le score optimal et affichant le résultat de l'alignement. À travers une série de tests sur des séquences de différentes longueurs, nous avons évalué la performance de l'algorithme en termes de temps d'exécution et de qualité des alignements obtenus.

Ensuite, nous nous sommes penchés sur l'alignement multiple de séquences, qui permet de comparer simultanément plus de deux séquences. Nous avons exploré différentes approches, notamment l'approche progressive, et avons implémenté un algorithme basé sur la notion de profil. Les résultats obtenus ont été présentés progressivement à mesure que chaque séquence était ajoutée à l'alignement. Nous avons également évalué la performance de cette méthode en fonction du nombre de séquences et de leur longueur.

Ce projet nous a permis de mieux comprendre l'importance de l'alignement de séquences en bio-informatique. Nous avons constaté que l'alignement par paire est efficace pour comparer deux séquences, tandis que l'alignement multiple est essentiel pour étudier les relations évolutives entre plusieurs séquences. Cependant, chaque méthode présente ses propres limites en termes de complexité computationnelle et de précision des résultats. Il est donc crucial de choisir la méthode appropriée en fonction des besoins spécifiques de l'analyse. Enfin, ce projet nous a également permis d'appréhender les défis et les opportunités dans le domaine de la bio-informatique, et de développer nos compétences en programmation et en analyse des données biologiques.

## 8. Annexe

Tableau récapitulatif des résultats :				
Taille seq 1	Taille seq 2	Taille globale	Temps d'exécution	Score d'alignement
10	10	20	0.001006	4
10	20	30	0.001045	7
20	20	40	0.000000	12
30	30	60	0.000996	21
50	50	100	0.001004	33
100	100	200	0.001004	91
200	200	400	0.002000	230
500	500	1000	0.003999	546
1000	1000	2000	0.009006	1196

Figure 29: Tableau explicatif de l'alignement par paire

### Tableau d'informations :

Taille séquence	Nombre de séquences	Temps dexécution
50	3	0.005997 secondes
50	4	0.007995 secondes
50	5	0.009996 secondes
50	6	0.013001 secondes
50	7	0.015674 secondes
50	8	0.020004 secondes
50	9	0.022006 secondes
50	10	0.023264 secondes

### Tableau d'informations :

Taille séquence	Nombre de séquences	Temps dexécution
100	3	0.006001 secondes
100	4	0.008819 secondes
100	5	0.010009 secondes
100	6	0.013003 secondes
100	7	0.017540 secondes
100	8	0.011006 secondes
100	9	0.023532 secondes
100	10	0.027161 secondes

Tableau d'informations :

Taille séquence	Nombre de séquences	Temps dexécution
150	3	0.005999 secondes
150	4	0.009017 secondes
150	5	0.011643 secondes
150	6	0.016126 secondes
150	7	0.018538 secondes
150	8	0.014998 secondes
150	9	0.020193 secondes
150	10	0.023035 secondes
150	11	0.034366 secondes

Tableau d'informations :

Taille séquence	Nombre de séquences	Temps dexécution
200	3	0.007002 secondes
200	4	0.011000 secondes
200	5	0.015999 secondes
200	6	0.020056 secondes
200	7	0.018024 secondes
200	8	0.026688 secondes
200	9	0.033043 secondes
200	10	0.038176 secondes

Tableau d'informations :

Taille séquence	Nombre de séquences	Temps dexécution
300	3	0.007563 secondes
300	4	0.011316 secondes
300	5	0.020013 secondes
300	6	0.021534 secondes
300	7	0.026227 secondes
300	8	0.033032 secondes
300	9	0.062521 secondes
300	10	0.040670 secondes

Figure 30: Tableaux explicatif de l'alignement - Approche progressif

Résultats :

Taille de la séquence	Nombre de séquences	Temps d'exécution
100	3	0.241544
100	4	0.326585
100	5	0.293261
100	6	0.340454
100	7	0.403734
100	8	0.387194
100	9	0.409754
100	10	0.463689

Résultats :

Taille de la séquence	Nombre de séquences	Temps d'exécution
150	3	0.285552
150	4	0.321794
150	5	0.418824
150	6	0.412699
150	7	0.442606
150	8	0.494291
150	9	0.509848
150	10	0.548218

Résultats :

Taille de la séquence	Nombre de séquences	Temps d'exécution
200	3	0.330859
200	4	0.406071
200	5	0.424856
200	6	0.498104
200	7	0.509691
200	8	0.617193
200	9	0.810479
200	10	0.633210

Voulez-vous continuer ? (o/n) n

Résultats :

Taille de la séquence	Nombre de séquences	Temps d'exécution
300	3	0.436182
300	4	0.511508
300	5	0.631581
300	6	0.652109
300	7	0.736005
300	8	0.769468
300	9	0.804323
300	10	0.913221

Figure 31: Tableaux explicatif de l'alignement - Approche itérative (SAGA)

## Code source des algorithmes

### Algorithme de Needleman et Wunsch

```
#!/usr/bin/env python3
import sys
import random
import time
# Fonction pour générer une séquence aléatoire d'acides aminés de longueur donnée
def generate_sequence(length):
    amino_acids = ['A', 'R', 'N', 'D', 'C', 'Q', 'E', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V']
    return ''.join(random.choices(amino_acids, k=length))
# Fonction pour aligner deux séquences d'acides aminés
def align_sequences():
    # Demander à l'utilisateur d'entrer la taille des deux séquences
    seq1_length = int(input("Entrez la taille de la première séquence : "))
    seq2_length = int(input("Entrez la taille de la deuxième séquence : "))
    # Générer les séquences aléatoires d'acides aminés
    seq1 = generate_sequence(seq1_length)
    seq2 = generate_sequence(seq2_length)
    # Afficher les séquences générées
    print("Séquence 1 :", seq1)
    print("Séquence 2 :", seq2)
    m = len(seq1) # Seq1 sera la séquence verticale à gauche
    n = len(seq2) # Seq2 sera la séquence horizontale en haut
    init_mat = [] # Matrice initialisée
    # Système de notation pour correspondance, non-correspondance et trou
    blosum62 = {
        ('A', 'A'): 4, ('A', 'R'): -1, ('A', 'N'): -2, ('A', 'D'): -2, ('A', 'C'): 0,
        ('A', 'Q'): -1, ('A', 'E'): -1, ('A', 'G'): 0, ('A', 'H'): -2, ('A', 'I'): -1,
        ('A', 'L'): -1, ('A', 'K'): -1, ('A', 'M'): -1, ('A', 'F'): -2, ('A', 'P'): -1,
        ('A', 'S'): 1, ('A', 'T'): 0, ('A', 'W'): -3, ('A', 'Y'): -2, ('A', 'V'): 0,
        ('R', 'R'): 5, ('R', 'N'): 0, ('R', 'D'): -2, ('R', 'C'): -3, ('R', 'Q'): 1,
        ('R', 'E'): 0, ('R', 'G'): -2, ('R', 'H'): 0, ('R', 'I'): -3, ('R', 'L'): -2,
        ('R', 'K'): 2, ('R', 'M'): -1, ('R', 'F'): -3, ('R', 'P'): -2, ('R', 'S'): -1,
        ('R', 'T'): -1, ('R', 'W'): -3, ('R', 'Y'): -2, ('R', 'V'): -3,
        ('N', 'N'): 6, ('N', 'D'): 1, ('N', 'C'): -3, ('N', 'Q'): 0, ('N', 'E'): 0,
        ('N', 'G'): 0, ('N', 'H'): 1, ('N', 'I'): -3, ('N', 'L'): -3, ('N', 'K'): 0,
        ('N', 'M'): -2, ('N', 'F'): -3, ('N', 'P'): -2, ('N', 'S'): 1, ('N', 'T'): 0,
        ('N', 'W'): -4, ('N', 'Y'): -2, ('N', 'V'): -3,
        ('D', 'D'): 6, ('D', 'Q'): -3, ('D', 'O'): 0, ('D', 'E'): 2, ('D', 'G'): -1,
        ('D', 'H'): 1, ('D', 'I'): -3, ('D', 'L'): -4, ('D', 'K'): -1, ('D', 'M'): -3,
        ('D', 'F'): -3, ('D', 'P'): -1, ('D', 'S'): 0, ('D', 'T'): -1, ('D', 'W'): -4,
        ('D', 'Y'): -3, ('D', 'V'): -3,
        ('C', 'C'): 9, ('C', 'Q'): -3, ('C', 'E'): -4, ('C', 'G'): -3, ('C', 'H'): -3, ('C', 'I'): -1, ('C', 'L'): -1, ('C', 'K'): -3, ('C', 'M'): -1,
        ('C', 'F'): -2, ('C', 'P'): -3, ('C', 'S'): -1, ('C', 'T'): -1, ('C', 'W'): -2,
        ('C', 'Y'): -2, ('C', 'V'): -1,
        ('Q', 'Q'): 5, ('Q', 'E'): 2, ('Q', 'G'): -2, ('Q', 'H'): 0, ('Q', 'I'): -3,
        ('Q', 'L'): -2, ('Q', 'K'): 1, ('Q', 'M'): 0, ('Q', 'F'): -3, ('Q', 'P'): -1,
        ('Q', 'S'): 0, ('Q', 'T'): -1, ('Q', 'W'): -2, ('Q', 'Y'): -1, ('Q', 'V'): -2,
        ('E', 'E'): 5, ('E', 'G'): -2, ('E', 'H'): 0, ('E', 'I'): -3, ('E', 'L'): -3,
        ('E', 'K'): 1, ('E', 'M'): -2, ('E', 'F'): -3, ('E', 'P'): -1, ('E', 'S'): 0,
        ('E', 'T'): -1, ('E', 'W'): -3, ('E', 'Y'): -2, ('E', 'V'): -2,
        ('G', 'G'): 6, ('G', 'H'): -2, ('G', 'I'): -4, ('G', 'L'): -4, ('G', 'K'): -2,
        ('G', 'M'): -3, ('G', 'F'): -3, ('G', 'P'): -2, ('G', 'S'): 0,
        ('G', 'T'): -2, ('G', 'W'): -2, ('G', 'Y'): -3, ('G', 'V'): -3,
        ('H', 'H'): 8, ('H', 'I'): -3, ('H', 'L'): -3, ('H', 'K'): -1,
        ('H', 'M'): -2, ('H', 'F'): -1, ('H', 'P'): -2, ('H', 'S'): -1,
        ('H', 'T'): -2, ('H', 'W'): -2, ('H', 'Y'): 2, ('H', 'V'): -3,
        ('I', 'I'): 4, ('I', 'L'): 2, ('I', 'K'): -3, ('I', 'M'): 1,
        ('I', 'F'): 0, ('I', 'P'): -3, ('I', 'S'): -2, ('I', 'T'): -1,
        ('I', 'W'): -3, ('I', 'Y'): -1, ('I', 'V'): 3,
        ('L', 'L'): 4, ('L', 'K'): -2, ('L', 'M'): 2, ('L', 'F'): 0,
        ('L', 'P'): -3, ('L', 'S'): -2, ('L', 'T'): -1, ('L', 'W'): -2,
        ('L', 'Y'): -1, ('L', 'V'): 1,
        ('K', 'K'): 5, ('K', 'M'): -1, ('K', 'F'): -3, ('K', 'P'): -1,
        ('K', 'S'): 0, ('K', 'T'): -1, ('K', 'W'): -3, ('K', 'Y'): -2,
        ('K', 'V'): -2,
        ('M', 'M'): 5, ('M', 'F'): 0, ('M', 'P'): -2, ('M', 'S'): -1,
        ('M', 'T'): -1, ('M', 'W'): -1, ('M', 'Y'): -1, ('M', 'V'): 1,
```

```

('F', 'F'): 6, ('F', 'P'): -4, ('F', 'S'): -2, ('F', 'T'): -2,
('F', 'W'): 1, ('F', 'Y'): 3, ('F', 'V'): -1,
('P', 'P'): 7, ('P', 'S'): -1, ('P', 'T'): -1, ('P', 'W'): -4,
('P', 'Y'): -3, ('P', 'V'): -2,
('S', 'S'): 4, ('S', 'T'): 1, ('S', 'W'): -3, ('S', 'Y'): -2,
('S', 'V'): -2,
('T', 'T'): 5, ('T', 'W'): -2, ('T', 'Y'): -2, ('T', 'V'): 0,
('W', 'W'): 11, ('W', 'Y'): 2, ('W', 'V'): -3,
('Y', 'Y'): 7, ('Y', 'V'): -1,
('V', 'V'): 4)

# Initialisation des scores de correspondance, de non-correspondance et de trou
match = 1
mismatch = -1
gap = -2

# Initialisation de la matrice à 0
init_mat = [[0] * (n + 1) for _ in range(m + 1)]

# Remplissage de la première ligne et de la première colonne de la matrice
for i in range(1, m + 1):
    init_mat[i][0] = gap * i

for j in range(1, n + 1):
    init_mat[0][j] = gap * j

# Remplissage de la matrice
for i in range(1, m + 1):
    for j in range(1, n + 1):
        match_score = blosum62.get((seq1[i - 1], seq2[j - 1]), mismatch)
        init_mat[i][j] = max(init_mat[i - 1][j - 1] + match_score,
                             init_mat[i - 1][j] + gap,
                             init_mat[i][j - 1] + gap)

# Temps de départ
start_time = time.time()

# Initialisation des séquences alignées
seq1_align = ""
seq2_align = ""

# Initialisation des indices de la matrice
i = m
j = n

# Backtracking
while i > 0 or j > 0:
    if i > 0 and j > 0 and init_mat[i][j] == init_mat[i - 1][j - 1] + blosum62.get((seq1[i - 1], seq2[j - 1]), mismatch):
        seq1_align = seq1[i - 1] + seq1_align
        seq2_align = seq2[j - 1] + seq2_align
        i -= 1
        j -= 1
    elif i > 0 and init_mat[i][j] == init_mat[i - 1][j] + gap:
        seq1_align = seq1[i - 1] + seq1_align
        seq2_align = '-' + seq2_align
        i -= 1
    else:
        seq1_align = '-' + seq1_align
        seq2_align = seq2[j - 1] + seq2_align
        j -= 1

# Calcul du score d'alignement
alignment_score = sum(blosum62.get((seq1_align[k], seq2_align[k]), mismatch) for k in range(len(seq1_align)))

# Temps d'exécution
execution_time = time.time() - start_time

# Affichage des résultats de l'alignement
print("Séquence 1 alignée:", seq1_align)
print("Séquence 2 alignée:", seq2_align)
print("Score d'alignement:", alignment_score)

```

```

print("Temps d'exécution:", execution_time, "secondes")

# Retourner les valeurs nécessaires dans un tuple
return seq1_length, seq2_length, seq1_length + seq2_length, execution_time, alignment_score

# Liste pour stocker les résultats des alignements
results = []

# Boucle pour permettre à l'utilisateur de continuer ou de quitter
while True:
    choice = input("Voulez-vous faire un alignement? (o pour continuer, n pour quitter) : ")
    if choice.lower() == 'o':
        result = align_sequences()
        results.append(result)
    elif choice.lower() == 'n':
        print("Programme terminé.")
        break
    else:
        print("Entrée invalide. Veuillez entrer 'o' pour continuer ou 'n' pour quitter.")

# Affichage du tableau récapitulatif des résultats
print("\nTableau récapitulatif des résultats :")
print("{:<12} {:<12} {:<16} {:<20} {:<16}".format("Taille seq 1", "Taille seq 2", "Taille globale", "Temps d'exécution", "Score d'alignement"))
for result in results:
    seq1_len, seq2_len, global_len, exec_time, alignment_score = result
    print("{:<12} {:<12} {:<16} {:<20.6f} {:<16}".format(seq1_len, seq2_len, global_len, exec_time, alignment_score))

# Affichage du tableau final des résultats dans le format spécifié
print("\nSequence 1 Sequence 2 Taille globale Temps d'exécution Score d'alignement")
for result in results:
    seq1_len, seq2_len, global_len, exec_time, alignment_score = result
    print("{:<12} {:<12} {:<16} {:<20.6f} {:<16}".format(seq1_len, seq2_len, global_len, exec_time, alignment_score))

```

## Sequence Alignment by Genetic Algorithm

```

import random
import time

# Générer une séquence aléatoire de nucléotides de longueur donnée
def generate_sequence(length):
    nucleotides = ['A', 'T', 'C', 'G']
    return ''.join(random.choices(nucleotides, k=length))

# Fonction pour évaluer un alignement (score simplifié)
def evaluate_alignment(alignment):
    score = 0
    for i in range(len(alignment[0])):
        column = [seq[i] for seq in alignment]
        if len(set(column)) == 1: # Tous les éléments de la colonne sont identiques

```

```

        score += 1
    return score

# Fonction pour aligner deux séquences en les complétant avec des gaps
def align_sequences(seq1, seq2):
    max_len = max(len(seq1), len(seq2))
    seq1 += '-' * (max_len - len(seq1))
    seq2 += '-' * (max_len - len(seq2))
    return seq1, seq2

# Fonction pour créer une population initiale d'alignements
def create_initial_population(sequences, population_size):
    population = []
    for _ in range(population_size):
        alignment = sequences[:]
        random.shuffle(alignment)
        population.append(alignment)
    return population

# Fonction pour effectuer une mutation sur un alignement
def mutate_alignment(alignment):
    idx = random.randint(0, len(alignment) - 1)
    seq = alignment[idx]
    if random.random() < 0.5:
        # Insertion d'un gap
        pos = random.randint(0, len(seq))
        alignment[idx] = seq[:pos] + '-' + seq[pos:]
    else:
        # Suppression d'un gap
        seq = seq.replace('-', '')
        alignment[idx] = seq
    # Ajuster toutes les séquences à la même longueur
    max_len = max(len(seq) for seq in alignment)
    for i in range(len(alignment)):
        alignment[i] += '-' * (max_len - len(alignment[i]))
    return alignment

# Fonction pour croiser deux alignements (croisement simple)
def crossover_alignment(parent1, parent2):
    if parent1 is None or parent2 is None:
        return None, None
    cross_point = random.randint(1, len(parent1[0]) - 1)
    child1 = [seq[:cross_point] + seq2[cross_point:] for seq, seq2 in zip(parent1, parent2)]
    child2 = [seq2[:cross_point] + seq[cross_point:] for seq, seq2 in zip(parent1, parent2)]
    # Ajuster toutes les séquences à la même longueur
    max_len = max(len(seq) for seq in child1 + child2)
    for i in range(len(child1)):
        child1[i] += '-' * (max_len - len(child1[i]))
        child2[i] += '-' * (max_len - len(child2[i]))
    return child1, child2

# Fonction pour la sélection des alignements (roulette wheel selection)
def select_alignment(population, scores):
    max_score = sum(scores)
    if max_score == 0:
        return random.choice(population)
    pick = random.uniform(0, max_score)
    current = 0
    for alignment, score in zip(population, scores):
        current += score
        if current > pick:
            return alignment
    return random.choice(population)

# Fonction principale pour l'algorithme SAGA
def saga_algorithm(sequences, population_size=10, generations=50):
    population = create_initial_population(sequences, population_size)
    best_alignment = None
    best_score = 0

    for generation in range(generations):

```

```

scores = [evaluate_alignment(alignment) for alignment in population]
next_generation = []

while len(next_generation) < population_size:
    parent1 = select_alignment(population, scores)
    parent2 = select_alignment(population, scores)
    if parent1 is None or parent2 is None:
        continue
    child1, child2 = crossover_alignment(parent1, parent2)
    if child1 is not None and child2 is not None:
        next_generation.append(mutate_alignment(child1))
        next_generation.append(mutate_alignment(child2))

population = next_generation[:population_size]

for alignment in population:
    score = evaluate_alignment(alignment)
    if score > best_score:
        best_score = score
        best_alignment = alignment

print(f"Génération {generation + 1}, meilleur score : {best_score}")

return best_alignment, best_score

def run_alignment_generation(seq_length, num_sequences):
    while True:
        # Générer les séquences aléatoires de nucléotides
        sequences = [generate_sequence(seq_length) for _ in range(num_sequences)]

        # Afficher les séquences générées
        print("\nSéquences générées :")
        for i, seq in enumerate(sequences, start=1):
            print(f"Séquence {i} : {seq}")

        # Exécuter l'algorithme SAGA
        start_time = time.time()
        best_alignment, best_score = saga_algorithm(sequences)
        end_time = time.time()
        execution_time = end_time - start_time

        # Afficher les résultats
        print("\nMeilleur alignement trouvé :")
        for seq in best_alignment:
            print(seq)
        print(f"\nScore de l'alignement : {best_score}")
        print(f"\nTemps total d'exécution : {execution_time:.6f} secondes")

        # Demander à l'utilisateur s'il souhaite continuer
        choice = input("Voulez-vous continuer ? (o/n)").lower()
        if choice != 'o':
            break

        # Demander à l'utilisateur d'entrer le nombre de séquences à nouveau
        num_sequences = int(input("Entrez le nombre de séquences :"))

    # Demander à l'utilisateur d'abord d'entrer la taille de chaque séquence
    seq_length = int(input("Entrez la taille de chaque séquence :"))

    # Demander à l'utilisateur d'entrer le nombre de séquences
    num_sequences = int(input("Entrez le nombre de séquences :"))

    # Exécuter l'alignement avec les paramètres donnés
    run_alignment_generation(seq_length, num_sequences)

```

## L'Approche Progressive

```

import random
import time

# Déclaration des variables globales pour stocker les informations
alignments_info = []

# Générer une séquence aléatoire de nucléotides de longueur donnée
def generate_sequence(length):
    nucleotides = ['A', 'T', 'C', 'G']
    return ''.join(random.choices(nucleotides, k=length))

# Demander à l'utilisateur d'entrer la taille des séquences
def get_sequence_length():
    seq_length = int(input("Entrez la taille de chaque séquence : "))
    return seq_length

# Demander à l'utilisateur d'entrer le nombre de séquences
def get_num_sequences():
    num_sequences = int(input("Entrez le nombre de séquences : "))
    return num_sequences

# Générer les séquences aléatoires de nucléotides
def generate_sequences(num_sequences, seq_length):
    sequences = [generate_sequence(seq_length) for _ in range(num_sequences)]
    return sequences

# Afficher les séquences générées
def display_sequences(sequences):
    print("Séquences générées :")
    for i, seq in enumerate(sequences, start=1):
        print(f"Séquence {i} : {seq}")

# Alignement simple de deux séquences
def align_sequences(seq1, seq2):
    aligned_seq1 = ""
    aligned_seq2 = ""
    for base1, base2 in zip(seq1, seq2):
        if base1 == base2:
            aligned_seq1 += base1
            aligned_seq2 += base2
        else:
            aligned_seq1 += "-"
            aligned_seq2 += "-"
    return aligned_seq1, aligned_seq2

# Alignement d'une séquence avec un profil
def align_with_profile(seq, profile):
    aligned_seq = ""
    for base, profile_base in zip(seq, profile):
        if base == profile_base or profile_base == '-':
            aligned_seq += base
        else:
            aligned_seq += "-"
    return aligned_seq

# Construction d'un profil à partir d'alignements
def build_profile(aligned_sequences):
    profile = ""
    for bases in zip(*aligned_sequences):
        base_counts = {}
        for base in bases:
            base_counts[base] = base_counts.get(base, 0) + 1
        most_common_base = max(base_counts, key=base_counts.get)
        profile += most_common_base
    return profile

# Construction d'une matrice de distances entre toutes les séquences
def build_distance_matrix(sequences):
    distance_matrix = []
    for i in range(len(sequences)):
        distances = []

```

```

for j in range(len(sequences)):
    if i != j:
        distance = sum(a != b for a, b in zip(sequences[i], sequences[j]))
        distances.append(distance)
    else:
        distances.append(0)
distance_matrix.append(distances)
return distance_matrix

# Détermination de l'ordre des alignements en fonction des distances
def determine_alignment_order(distance_matrix):
    order = []
    remaining_sequences = set(range(len(distance_matrix)))
    while remaining_sequences:
        min_distance = float('inf')
        closest_pair = None
        for i in remaining_sequences:
            for j in remaining_sequences:
                if i != j and distance_matrix[i][j] < min_distance:
                    min_distance = distance_matrix[i][j]
                    closest_pair = (i, j)
        order.append(closest_pair)
        remaining_sequences -= set(closest_pair)
    return order

def progressive_alignment(sequences):
    aligned_sequences = [sequences[0]]
    profiles = [sequences[0]] # Initialiser le premier profil avec la première séquence
    start_time = time.time() # Commencer à mesurer le temps d'exécution
    for i in range(1, len(sequences)):
        seq_to_align = sequences[i]
        profile = build_profile(aligned_sequences)
        aligned_seq = align_with_profile(seq_to_align, profile)
        aligned_sequences.append(aligned_seq)
        profiles.append(profile)
        print("\nAlignement progressif étape", i, ":")
        print(f"Séquence {i + 1} alignée : {aligned_seq}")
        print(f"Profil {i + 1} : {profile}")

    end_time = time.time() # Finir de mesurer le temps d'exécution
    execution_time = end_time - start_time # Calculer le temps total d'exécution
    alignments_info.append((len(sequences[0]), len(sequences), execution_time)) # Stocker les informations de l'alignement
    print(f"Temps total d'exécution : {execution_time:.6f} secondes") # Afficher le temps total d'exécution

# Boucle principale pour permettre à l'utilisateur de faire plusieurs alignements
while True:
    print("\nNouvel alignement :")
    seq_length = get_sequence_length() # Demander la taille des séquences
    num_sequences = get_num_sequences() # Demander le nombre de séquences
    sequences = generate_sequences(num_sequences, seq_length)
    display_sequences(sequences)
    progressive_alignment(sequences)
    choice = input("Voulez-vous faire un autre alignement? (o pour continuer, n pour quitter) : ")
    if choice.lower() != 'o':
        print("Programme terminé.")
        break

# Affichage du tableau d'informations
print("\nTableau d'informations :")
print(f"\n{'Taille séquence':<20}{{'Nombre de séquences':<20}{'Temps dexécution':<20}}")
for info in alignments_info:
    print(f"\n{info[0]:<20}{info[1]:<20}{info[2]:.6f} secondes")

```