الجمهورية الجزائرية الديمقراطية الشعبية

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Frères Mentouri Constantine 1 — **جامعة الإخوة منتوري قسنطينة 1**

Faculté des Sciences de la Nature et de la Vie — **كلية علوم الطبيعة والحياة**

Département : Biologie Appliquée — ***قسم البيولوجيا التطبيقية***

Mémoire

Présenté en vue de l'obtention du Diplôme de Master
Filière : Sciences biologiques
Spécialité : Bioinformatique.

Intitulé

Implication des méthodes de l'intelligence artificielle dans l'alignement des séquences

❖ **Présenté et soutenu :**
- **Le :**
- **Par :** BENOTMANE Meriem
  CHEIKH BOUKAL Selma

❖ **Jury d'évaluation :**
- Président de jury : Dr.BELLIL Ines — MCA. UFM. Constantine 1.
- Encadreur : Dr.DAAS Mohamed Skander — MCA. UFM. Constantine 1.
- Examinateur : Dr.DJOUDI Brahim — MCB. UFM. Constantine 1.

- Année universitaire : 2020–2021 –

Remerciements

Avant tout nous remercions Dieu le tout-puissant qui nous a donné la force et le courage pour qu'on puisse accomplir ce modeste travail.

Ce travaille a été réalisé grâce a la collaboration directe ou indirecte de plusieurs personnes qui per leurs suggestion et leurs connaissances ont permis l'aboutissement de ce travail

Mes vifs remerciement à ***Dr DAAS Mohamed Skander*** *mon encadreur pour avoir accepté de diriger ce travail avec compétence et professionnalisme*

Nous remercions aussi les membres de jury ***Dr.BELLIL Ines*** *et*

Dr.DJOUDI Brahim

de nous faire l'homme de présider le jury de notre mémoire.

Merci à tous

Dédicace

Je dédie cet humble travail à tous les êtres qui me sont chères

A l'âme de mon regrettable défunt père ***« Moukhtar »***

A l'âme de mon chère défunt frère ***« Amer »***

J'aurais aimé qu'il soient présent avec mois pendant cette période cruciale de ma vie

A la source de mon bonheur et ma force, ma tendre adorable maman ***« Fatima Zohra »****pour son amour, sa patience ,ses sacrifices et ses prières qui mon accompagnées toute ma vie*

A ma gentille sœur ***« Hiba »****et sa petite famille pour son soutient et ses conseils précieux*

*A mon cher frére****« Abd el Aziz »****pour son aide et ses encouragement*

A mes belles sœurs ***« Ibtissem »****et* ***« Achouak »***

A mes petits amours ***« Djanna »*** *et* ***« Amer »***

A toute la famille ***« Benotmane »****et la famille* ***« Brihmat »***

je cite spécialement mon oncle ***« Mohamed lamine Brihmat »****et ma tante* ***« Ilham Brihmat »***

Aucune dédicace ne saurait exprimer mon respect et mon affection à vous tous.

Meriem

Dédicaces

Je dédie ce présent mémoire et ouvrage

A l'unique et seule personne que j'ai toujours aimé et respecté, à man très cher remarquable père ***« Papa »***

A man adorable, la seule raison et espoir de vivre , ma très chère ***« mama »,*** *ma première et dernière école.*

A man mari ***« Salim »*** *que je respecte et à qui je réserve une grand entente et amour à sa famille* ***Père*** *et* ***Mère*** *et* ***sœurs***

A mes fréres, ***Issam*** *;****Imed*** *;****Mouncef*** *et* ***Youcef*** *à qui je leurs souhaite bonne réussite dans la vie.*

A mes adorables sœurs, ***Sara*** *et* ***Feriel*** *que je considère mes seules amies et copines et à qui je me confie*

A ma grand-mère paternelle à l'unique Amouta, dite ***Meriem***.

A man binôme Meriem ;pour se collaboration et sa compréhension

A tous ceux qui m'ont aidé à réaliser et accomplir cet mémoire

selma

TABLE DES MATIERES

CHAPITRE 2 : Intelligence artificielle et son application a l'alignement de séquences

Partie pratique

CHAPITRE 3 : Utilisation et évaluation des algorithmes génétiques pour l'alignement multiple de séquences

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

MSA : Multiple Séquence Alignement.

PAM : Mutation Ponctuelles Acceptées.

BLOSUM : Matrice de substitution de blocs.

DCA: Algorithme Diviser pour régner.

WSP : Fonction objectif.

SAGA : Système pour les Analyses Géoscientifiques Automatisées.

N.J : Neighbor Joining.

UPGMA : Méthode des Groups de Paires non pondérées avec Moyenne Arithmétique.

IA : Intelligence artificielle.

S.E : Systèmes Experts.

ACO : L'optimisation des colonies de fourmis et l'algorithme génétique simple.

RBT : Registered Betravior Technician.

GA : Algorithmegénétiqu

Introduction Générale

Introduction Générale

La disponibilité d'une grande quantité d'information sur les séquences d'ADN et en particulier sur les génomes complets de plus de 200 espèces, a ouvert une nouvelle ère dans l'histoire de la biologie moléculaire. Les banques de données dédiées connaissent une croissance rapide et fructueuse. Les ambitions des biologistes et leur curiosité augmentent au même rythme que les découvert biologiques se succèdent.

Les souhaits d'un biologiste actuel sont :

1. Analyser, comprendre et organiser une masse de données biologiques
2. Décoder l'information contenue dans les séquences d'ADN et de protéines
3. Modéliser les structures 3D des protéines et des ARNs structurels et déterminer la relation entre structure et fonction
4. Étudier la régulation des gènes et déterminer les réseaux d'interaction entre les protéines.

La bioinformatique est une branche de biologie se développe rapidement et qui est fortement interdisciplinaire , en utilisant des techniques et des concepts de l'informatique, des statistiques, des mathématiques, de la chimie, de la biochimie, de la physique, et de la linguistique. La bioinformatique extrait et interprète la connaissance de l'analyse par ordinateur des données biologiques. Celles-ci peuvent comprendre l'information stockée dans le code génétique, mais également des résultats expérimentaux de diverses sources, des statistiques médicales, et la littérature scientifique. La recherche en bioinformatique inclut le développement de méthodes pour le stockage, la récupération, et l'analyse des données. Il existe abondamment d'applications pratiques dans différents secteurs de biologie et de médecine.

L'alignement des séquences d'ADN ou de protéines est une des techniques les plus utilisées dans l'analyse de séquence. Il est considéré parmi les problèmes les plus difficiles en bioinformatique. L'alignement de séquences est une tâche cruciale et très importante en biologie moléculaire.

Dans ce mémoire nous étudions l'apport de l'intelligence artificielle (IA), qui englobe plusieurs méthodes, au problème de l'alignement de séquences. La première partie de ce mémoire est constituée de deux chapitres. Nous présentons les concepts de base sur l'alignement de séquences et plusieurs méthodes d'alignement dans le premier chapitre. Le

deuxième chapitre présente l'intelligence artificielle et décrit principalement ses techniques. La deuxième partie de ce mémoire présentée par le chapitre 3 est consacrée à expliquer notre travail qui consiste à étudier la performance d'un algorithme génétique qui appartient aux méthodes de l'IA et plus précisément aux métaheuristiques. Cette étude comparera l'approche considérée avec une autre très répondue pour le problème d'alignement de séquences. Dans la dernière partie, une conclusion résume l'ensemble du travail et des perspectives.

Partie théorique

CHAPITRE 1 :

Alignement de séquences

CHAPITRE 1 : Alignement de séquences

1-Introduction:

La biologie moléculaire est l'étude des processus de réplication, de transcription et de traduction du matériel génétique. Le dogme central de la biologie moléculaire où le matériel génétique est transcrit en ARN, puis traduit en protéines, bien qu'il soit une image très simpliste et sans fondement de la biologie moléculaire. L'essentiel du travail en biologie moléculaire est quantitatif, et récemment beaucoup de travaux ont été faits à l'intersection de la biologie moléculaire et de l'informatique, dans la bio-informatique et dans la biologie calculatoire. Depuis les années 2000, l'étude de la structure et de la fonction des gènes, la génétique moléculaire, fait partie des sous-domaines les plus saillants de la biologie moléculaire**[1].**

Les biologistes utilisent l'alignement pour comparer de nouvelles séquences d'ADN à des séquences déjà étudiées. Cela leur permet de déterminer les séquences les plus proches et avoir ainsi une idée sur la fonction de la nouvelle séquence.

L'alignement de séquences est un autre problème qui est étudié depuis plus de quarante ans. il est utilisé dans beaucoup de problèmes en bioinformatique : identifier les sites fonctionnels, prédire la fonction d'une protéine, prédire la structure secondaire ou tertiaire d'une protéine ou d'un ARN, établir une phylogénie**[2].**

2-Représentation de l'alignement:

L'alignement de séquences est une manière de représenter deux ou plusieurs séquences (nucléotides, acides aminés) les unes sous les autres, de manière à en faire ressortir les régions homologues ou similaires (les zones de concordance). Elle nous permet de révéler trois opérations d'édition: 1) Substitution 2) Délétion 3) Insertion**[3].**

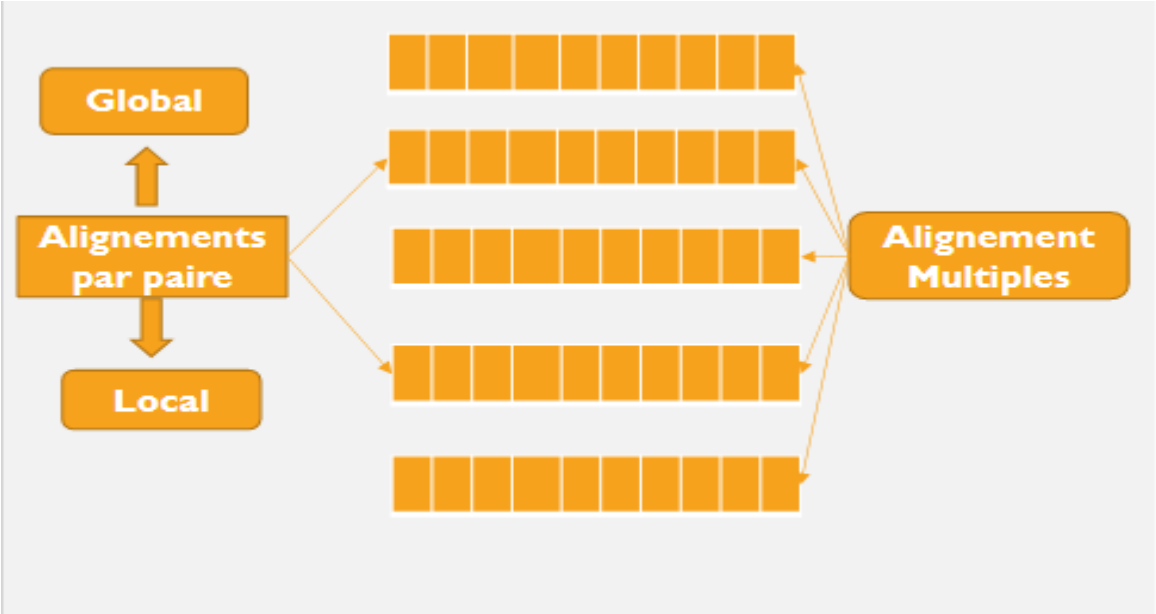


Figure 1 : Schéma d'un alignement de séquences**[9].**

Exemple:

Soit les deux séquences suivantes:

Sequence 1: A G V S I L N Y A

Sequence 2: V S I L Y A K R

1	2	3	4	5	6	7	8	9
A	G	V	S	I	L	N	Y	A
V	S	I	G	Y	A	K	R	

Si nous admettons que la séquence 2 a perdu les deux acides aminés N terminaux au cours de l'évolution, l'alignement devient:

1	2	3	4	5	6	7	8	9	10
A	G	V	S	I	L	N	Y	A	---
---	---	V	S	I	G	Y	A	K	R

3-Alignement de deux séquences et alignement multiple:

- **Alignement de deux séquences :**

Un alignement de deux séquences (appelé souvent '*Alignement deux à deux'*) est une mise en correspondance entre les résidus avec une possible insertion des espaces (gaps) afin d'obtenir des séquences de longueurs égales. Toutes les correspondances sont autorisées à condition que l'ordre des résidus soit respecté**[4].**

Trois situations sont possibles pour une position donnée de l'alignement :

- Les caractères sont les mêmes : identité

- Les caractères ne sont pas les mêmes : Substitution

- L'une des positions est un gap (espace) : Insertion/Délétion

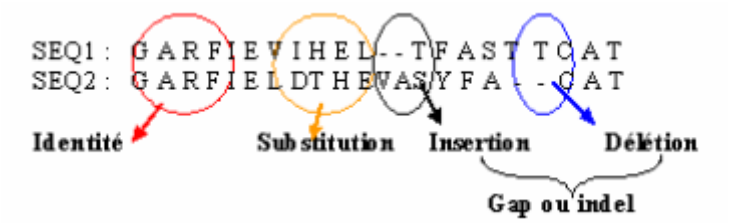


Figure 2 : Alignement de deux séquences protéiques**[4].**

- **Alignement multiple de séquences :**

Un alignement de séquences multiples (MSA: multiple séquence alignement) est un alignement de séquences de trois ou plusieurs séquences biologiques, généralement des protéines de l'ADN ou de l'ARN. Il permet de mettre en évidence les relations entre les séquences que l'on ne peut visualiser en comparant les séquences deux à deux**[4].**

Ce type d'alignement est appliqué surtout pour :

- La caractérisation des régions (motifs, domaines) conservées des protéines.

- Prédiction des structures 2D ou 3D par comparaison avec des structures connues.

- Construction des arbres phylogénétiques des séquences homologues.

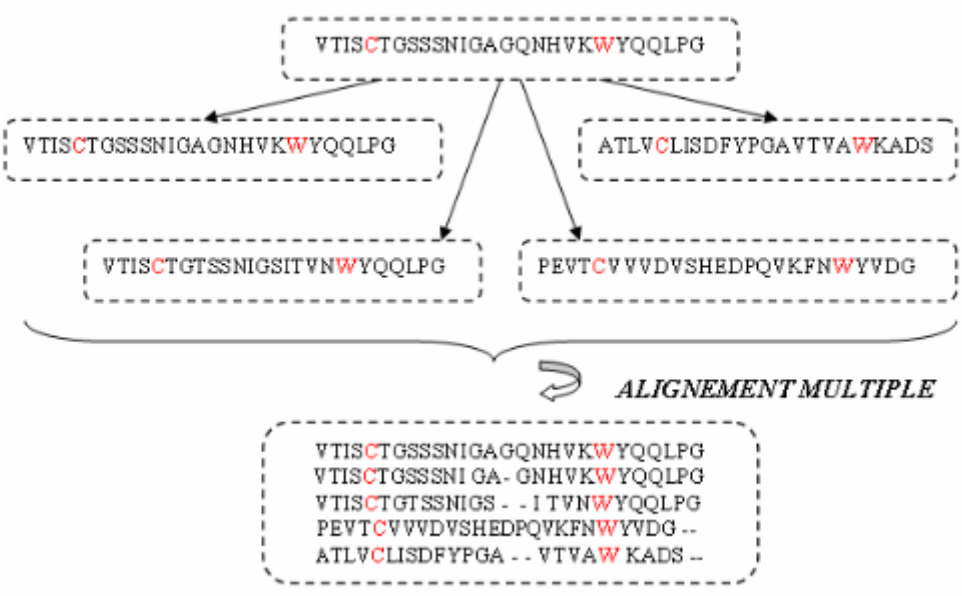


Figure 3 : Alignement multiple de séquences**[4].**

4 - Alignement global et alignement local

L'alignement global et l'alignement local sont toujours effectués entre deux séquences complètes sous des conditions différentes.

L'alignement global lorsque d'aligner tous les résidus des deux séquences, d'identifier les régions conservées et les différences entre ces deux séquences. L'algorithme le plus connu pour l'alignement global est l'algorithme de Needleman-Wunsch. Le score de similitude est calculé sur l'alignement complet.

Exemple : Un alignement global de score maximal entre deux séquences arbitraires dont les séquences sont : NLGPSTKDFGKISREFDNQ et LERSFGKINMRLEDA. La matrice utilisée est BLOSUM 62, les pénalités d'ouverture et d'extension d'un gap sont 10 et 0.5. L'alignement est généré par le serveur web "EMBOSS Needle du site EBI"**[5].**

L'alignement local lorsque de voir si une région dans une séquence s'aligne bien avec une région dans l'autre séquence. Il est utilisé quand Les séquences sont dissemblables et susceptibles de contenir des régions ou des motifs similaires. L'algorithme le plus connu pour l'alignement local des séquences est l'algorithme de Smith-Waterman [**5**].

Exemple : Un alignement local de score maximal entre deux séquences arbitraires : NLGPSTKDFGKISREFDNQ et LERSFGKINMRLEDA , la matrice utilisée est BLOSUM 62, les pénalité d'ouverture et d'extension d'un gap sont 10 et 0.5. L'alignement est généré

par le serveur web“EMBOSS Needle” La partie précédente nous permet de calculer le score de n’importe quel alignement deux-à deux, ce qui nous reste est de trouver l’alignement

optimal avec le score maximal. Nous pouvons utiliser, par exemple, les algorithmes de type programmation dynamique, comme l’algorithme de Needleman-Wunsch pour l’alignement global, celui Smith-Waterman pour l’alignement local et celui de Gotoh pour l’alignement global ou local en utilisant la pénalité affine de gap**[5].**

5-Fonction de score de similarité:

Exemple [6]:

G T T A A G G C G – G G A A A

G T T – – – G C G A GG A C A

* * * * * * * * *

Score = Score Identités + Score Différences

Identité= +1

Substitution=0

Gap= -1

Score=10-4=6

5.1-Cas des acides aminés:

• Plus difficile à modéliser que celui des nucléotides :

– Un acide aminé peut être remplacé par un autre de différentes façons (code génétique) :

• Asp (GAC) ⟶ Tyr (UAC, UAU) 1 ou 2 mutations

– Le nombre de substitutions requises pour passer d’un acide aminé à un autre diffère :

Asp (GAC, GAU) ⟶Tyr (UAC, UAU) 1 mutation

Asp (GAC, GAU) ⟶ Cys (UGC, UGU) 2 mutations

Asp (GAC, GAU) ⟶Trp (UGG) 3 mutations

– La probabilité des substitutions au niveau nucléotidique diffère :

P(AAUAsn|GAUAsp) > P(AAUAsn|CAUHis)

– Certaines substitutions peuvent avoir plus ou moins d’effet sur la fonction des protéines.

• Acidité, hydrophobicité, structure des protéines, etc**[6].**

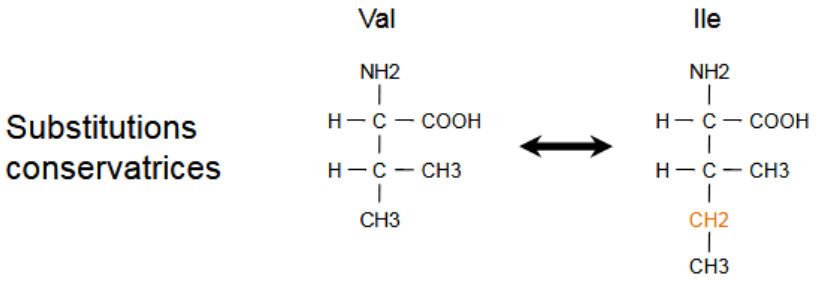
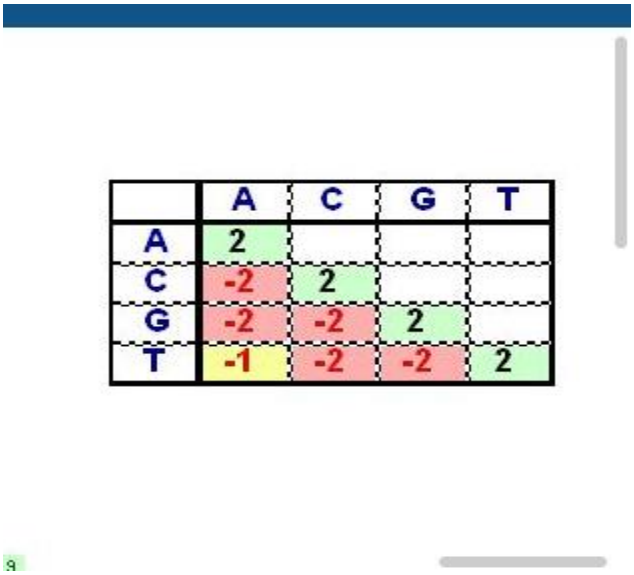


Figure N° 4 : structure des protéines**[6].**

6-Matrice de substitution:

- Une matrice de substitution associe un score à chaque paire de résidus qu'on peut trouver dans un alignement.



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

Figure N° 5 : Exemple d'une matrice de substitution**[7].**

- Chaque ligne et chaque colonne représente l'un des résidus (4 nucléotides, 20 acide aminés).
- La diagonale correspond aux identités.
- Le triangle inférieur correspond à des substitutions.
- Le triangle supérieur est symétrique au triangle inférieur, il n'est pas nécessaire d'indiquer les nombres.
- Les scores négatifs sont considérés comme des pénalités associées à certaines substitutions qu'on n'observe que rarement dans les alignements. Les algorithmes d'alignements tenteront donc d'éviter ces substitutions.

- Les scores positifs correspondent à des substitutions qu'on observe plus souvent que prévu, dans les alignements d'un grand nombre de séquences. Ceci suggère que ces substitutions particulières sont moins dommageables que d'autres, et on les qualifie donc de « substitutions conservatives » ou encore de « mutations ponctuelles acceptées » (PAM).
- Au sein d'un alignement, le terme similarité désigne les positions où se superposent des résidus ayant un score positif dans la matrice de substitution (identité ou substitution conservative). **[7]**

-Nombreuses matrices de substitution ont été proposées :

- Matrices de substitution BLOSUM 62:

Les matrices BLOSUM appartiennent à la série de matrices de substitution la plus couramment utilisée pour la recherche d'homologie de protéines et les alignements de séquences depuis leur publication en 1992. En 2008**[8]**, Styczynski et al. découvert des erreurs de calcul lors de l'étape de regroupement du calcul matriciel**[8].**

		Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
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 6 : Matrice de substitution BLOSUM 62**[9]**.

- Matrices de substitution PAM 250 :

Elles ont été créées par Magaret Dayhoff**[9]** et ses collaborateurs, après l'alignement d'environ 1300 séquences très semblables (> 85% d'identité) appartenant à 71 familles de protéines.

Les matrices PAM donnent 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"**[9].**

	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	4

Figure 7 : Matrice de substitution PAM 250**[9].**

-Le tableau suivant est une comparaison des modes de calculs des valeurs des matrices PAM et BLOSUM.

Tableau N° 1 : Comparaison des modes de calculs des valeurs des matrices PAM et BLOSUM.

Liens et différences entre PAM et BLOSUM [9]		
	PAM	**BLOSUM**
Pour comparer des séquences étroitement liées	des matrices avec des nombres plus faibles sont créées	des matrices avec des nombres plus élevés sont créées

Pour comparer des protéines distantes	des matrices avec des nombres élevés sont créées	des matrices avec des nombres faibles sont créées
Matrice	basée sur des alignements globaux de protéines étroitement apparentées	basée sur des alignements locaux
Un indice plus élevé dans la dénomination de la matrice reflète	une distance évolutive plus grande	une similarité de séquences plus élevée et donc une distance d'évolution plus petite
	PAM1 est la matrice calculée à partir de comparaisons de séquences n'ayant pas plus de 15% de divergence mais correspondant à 99% d'identité de séquence.	BLOSUM 62 est la matrice calculée à partir de comparaisons de séquences avec une identité par paire non supérieure à 62%.
	d'autres matrices PAM sont calculées à partir de PAM1	basé sur les alignements observés : ceux-ci ne sont pas extrapolés à partir de comparaisons de protéines étroitement apparentées
	Le score indique le pourcentage de substitution par poste	Le score indique le pourcentage de conservation

	Des nombres plus élevés sont appropriés pour des protéines plus éloignées	Des nombres plus élevés sont appropriés pour des protéines plus conservées

7-Pondération des gaps:

Avec ce système, une longue insertion est légèrement plus pénalisante qu'une courte, ce qui revient en fait à minimiser l'introduction même d'une insertion. Autrement dit, on facilitera souvent dans un alignement le fait d'avoir peu d'insertions, éventuellement longues, plutôt que d'avoir beaucoup d'insertions d'un seul élément. Ceci est tout à fait en concordance avec les événements biologiques observés car il peut se produire par exemple une seule délétion de plusieurs bases plutôt que plusieurs pertes indépendantes d'une seule base**[12].**

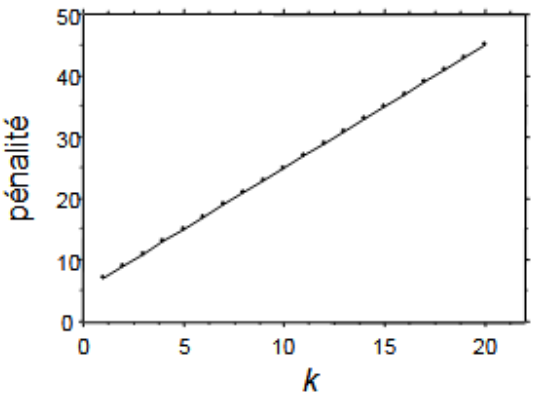
Pénalités linéaires :

$w = \delta_o + \delta_e k$

δ_o: pénalité pour l'ouverture d'un gap.

δ_e: pénalité pour l'extension d'un gap.

k : longueur du gap. **[6]**



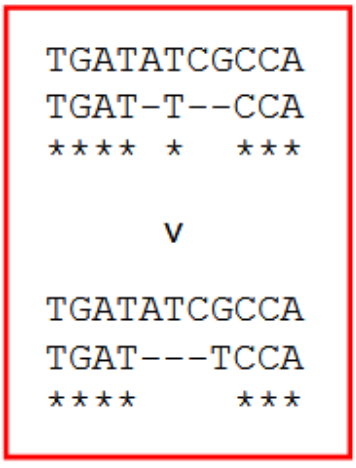


Figure N°8 : Pondération des gaps**[6].**

8-Quelques méthodes d'alignement

8.1-Méthodes exactes d'alignement multiple

8.1.1-Méthode MSA

C’est une tentative de rendre les algorithmes de la programmation dynamique conçus pour aligner deux séquences, opérationnels pour un alignement multiple. Sachant que la complexité temporelle et spatiale augmente proportionnellement avec le nombre et la longueur des séquences à aligner. Même la matrice de score devient elle aussi multidimensionnelle. Le programme de MSA emploie un algorithme intelligent pour réduire le volume de la matrice de la programmation dynamique multidimensionnelle. L'algorithme de Carrillo et deLipman **[10]** était mis en application dans MSA.

Le score d'un alignement multiple généré par une heuristique est la somme des scores de tous alignements deux à deux définis pour l'alignement multiple. Sachant que :

- Le score SP pour toute paire de séquences extraite de l’alignement multiple optimal, devrait être inférieur au score SP optimal de l’alignement de paires de séquences.
- Le score SP total d’un alignement optimal devrait être plus grand que celui d’un alignement obtenu par des méthodes heuristiques.

En plaçant la limite inférieure et la limite supérieure, seulement un espace restreint doit être exploré dans la table de score multidimensionnelle. Toutes ces considérations ont participé à la réduction du temps de calcul d’une manière significative. 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 séquences 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 2n Ln.
- Un espace mémoire proportionnel à Ln.
- Le problème de l’alignement multiple exacte a été démontré être un problème NP-complet.
- D’où Le recours aux méthodes approchées ou heuristiques**.**

8.1.2-Méthode de DCA

DCA (Divide and Conquer Algorithm) **[10]**, c’est une heuristique basée sur l’idée « diviser puis conquérir ». Le principe consiste à découper les séquences à aligner en sous-ensembles de segments. Ces segments doivent avoir une taille assez petite pour faciliter leur traitement par MSA. Les sous alignements produits sont alors concaténés pour former un seul

alignement multiple final. Comme étant une méthode exacte, elle hérite des mêmes inconvénients des méthodes de ce type : la complexité temporelle et spatiale**.**

8.2-Méthodes d'alignements Itératives

8.2.1- Méthode SAGA

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 **[10].**

Tableau N° 2 : Tableaux d'algorithm de SAGA.

Algorithm 2.1: SAGA	
Initialisation	
Evaluation:	1. create G0
	2. evaluate the population of generation n (Gn)
	3. if the population is stabilised then END
Breeding:	4. select the individuals to replace
	5. evaluate the expected offspring (EO)
	6. select the parent(s) from Gn
	7. select the operator
	8. generate the new child
	9. keep or discard the new child in Gn+1
	10. goto 6 until all the children have been successfully put into Gn+1
	11. n = n+1
	12. goto Evaluation
End :	13. end

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 ».

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

SAGA a la particularité de pouvoir optimiser n'importe quelle fonction objectif**.[17]**

8.2.2- Méthode DIALIG

DIALIGN est une méthode pour l'alignement multiple développée par Morgenstern . L'algorithme de DIALIGN est basé sur les alignements par paires de séquences (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 délétion 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. Dialign-t est une version plus récente de Dialign-2, locale et progressive**.[18]**

8.3- Les Méthodes d’alignements multiple progressives

8.3.1- Méthodes ClustalW

ClustalW **[25]** 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. 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**[10].**

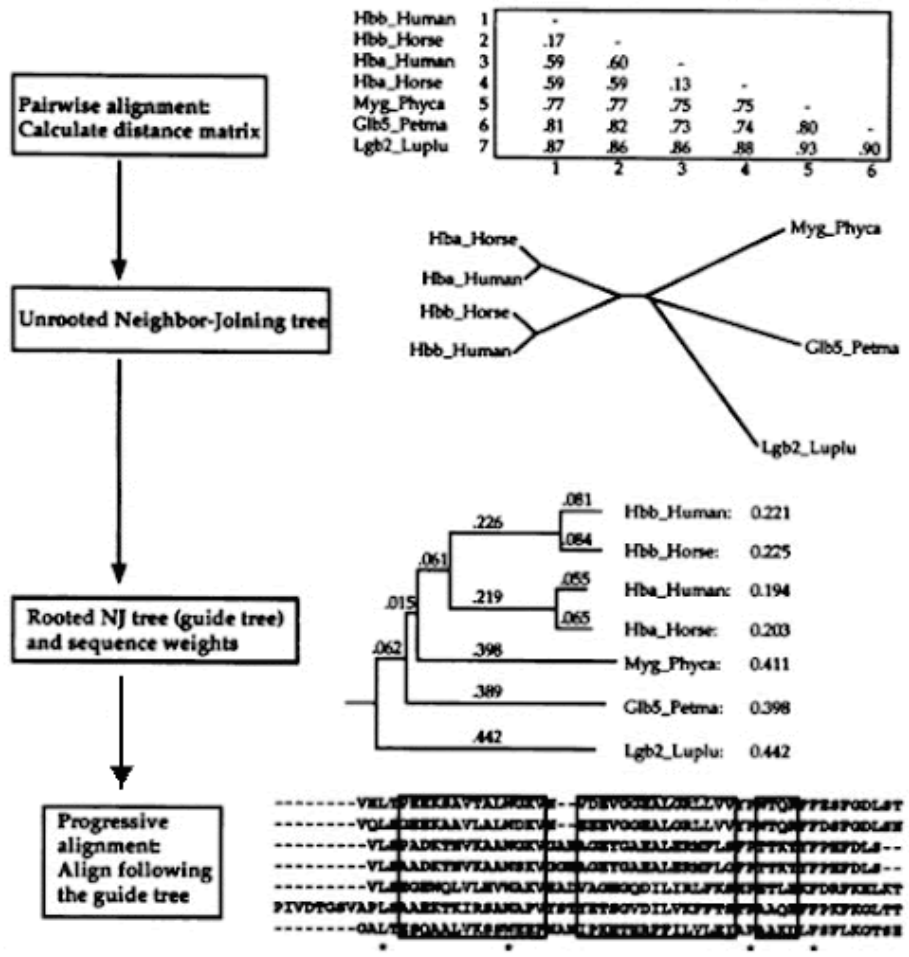


Figure 9 : Déroulement de l'algorithme ClustalW**[4].**

8.3.2- Méthodes MUSCLE

La méthode MUSCLE emploie deux mesures de distance pour une paire de séquences : une distance de k-mer (pour une paire non alignée) et le Kimura distance (pour une paire alignée). Un k-mer est une sous-séquence contiguë de longueur k également connu sous le nom de mot ou k-tuplet. Les séquences homogènes possèdent plus de k-mers en commun que prévu par hasard. Cette mesure n'exige pas un alignement, elle donne un avantage significatif de vitesse contrairement à Kimura.

La méthode MUSCLE peut être décrite en trois étapes essentielles :

Etape 1 : Le but de la première étape est de produire rapidement un alignement multiple avec plus d'exactitude possible. Ceci est basé sur la détermination d'une matrice D1 de distances à partir de la distance de k-mers entre toutes les paires de séquences. La matrice obtenue est alors clustérisée par UPGMA, pour produire un arbre binaire TREE1. Un alignement progressif MSA1 est construit alors en suivant l'ordre dicté par l'arbre.

Etape 2 : La source d'erreur principale à l'étape progressive est la mesure approximative de distances k-mer, qui a comme conséquence un arbre sous optimal. MUSCLE re-estime donc l'arbre en utilisant la distance de Kimura, qui est plus précise mais exige l'utilisation un alignement dans ce cas c'est MSA1 donnant la matrice D2. D2 va subir le même procédé de clustérisation afin de produire un arbre binaire TRRE2 et progressivement construire l'alignement MSA2.

Etape 3 : C'est une étape d'amélioration. TREE2 est divisé en deux sous arbres en

supprimant la branche qui les relient. Celle-ci est choisie en parcourant l'arbre à partir de la racine. Le profil de l'alignement multiple dans chaque sous arbre est alors calculé. Un nouvel alignement multiple est produit en réalignant les deux profils. Si le score de PS est amélioré, le nouvel alignement est gardé, autrement il est rejeté et l'étape 3 est alors répétée jusqu' à la convergence ou jusqu'à ce que une limite définie soit atteinte.

Considérée la plus rapide et plus exacte, la méthode MUSCLE est la plus répandue

actuellement avec ClustalW**[10].**

CHAPITRE 2 :

Intelligence artificielle et son application à l'alignement de

1-Introduction:

Depuis quelques années, l'Intelligence Artificielle (IA) fait l'objet d'une médiatisation et d'une attention sans précédent et suscite beaucoup de promesses, mais aussi des craintes dont certaines reposent sur des visions très spéculatives ou très lointaines des capacités des machines. Ce fort regain d'intérêt pour l'IA est notamment lié à d'importantes avancées technologiques qui ont permis d'accroître de façon considérable les performances des ordinateurs dans de nombreux domaines comme la reconnaissance automatique de la parole ou la vision par ordinateur. Ces avancées ont ouvert de vastes perspectives d'introduction de l'IA sous différentes formes (applications, robots, chatbots, etc.) dans les situations de travail. Un point particulièrement notable est que de plus en plus de secteurs sont concernés (industrie, santé, agriculture, finance, banque, assurance, transport, etc.). L'IA est ainsi sur le point de prendre une place de plus en plus importante dans les organisations et les systèmes de production. Les champs d'application de l'IA dans ces secteurs ne cessent de se multiplier (automatisation de tâches, relation client, logistique, analyse prédictive, diagnostic, analyse de grandes bases de données, etc.) **[11]**. L'alignement de séquences est considéré comme un problème difficile à traiter par les méthodes classiques. Où plusieurs méthodes montrent leurs limites face aux grand nombre de séquences à aligner et à ses longues tailles. Les méthodes de l'intelligence artificielle peuvent être une bonne alternative pour traiter ce problème.

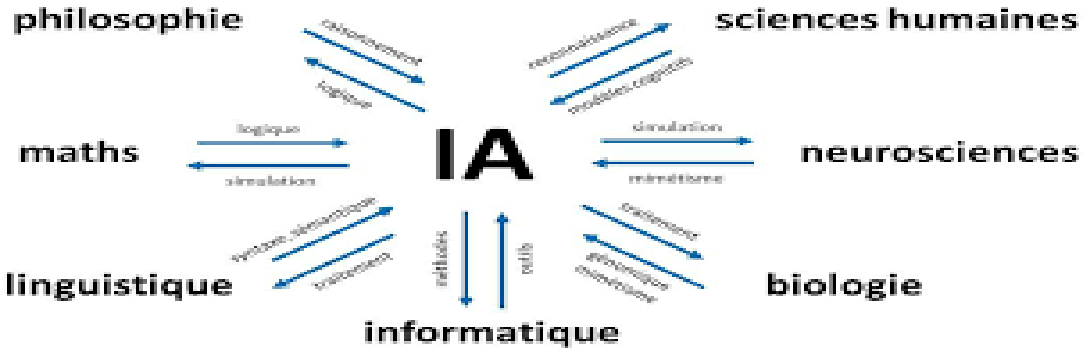


Figure 10 : Usages de l'intelligence artificielle**[12]**.

2- l'Intelligence artificielle:

L'intelligence artificielle (IA, ou AI en anglais pour *Artificial Intelligence*) consiste à mettre en œuvre un certain nombre de techniques visant à permettre aux machines d'imiter une forme d'intelligence réelle. L'IA se retrouve implémentée dans un nombre grandissant de domaines d'application. La notion voit le jour dans les années 1950 grâce au mathématicien Alan Turing. Dans son livre *Computing Machinery and Intelligence*, ce dernier soulève la question

d'apporter aux machines une forme d'intelligence. Il décrit alors un test aujourd'hui connu sous le nom « Test de Turing » dans lequel un sujet interagit à l'aveugle avec un autre humain, puis avec une machine programmée pour formuler des réponses sensées. Si le sujet n'est pas capable de faire la différence, alors la machine a réussi le test et, selon l'auteur, peut véritablement être considérée comme « intelligente »**[13].**

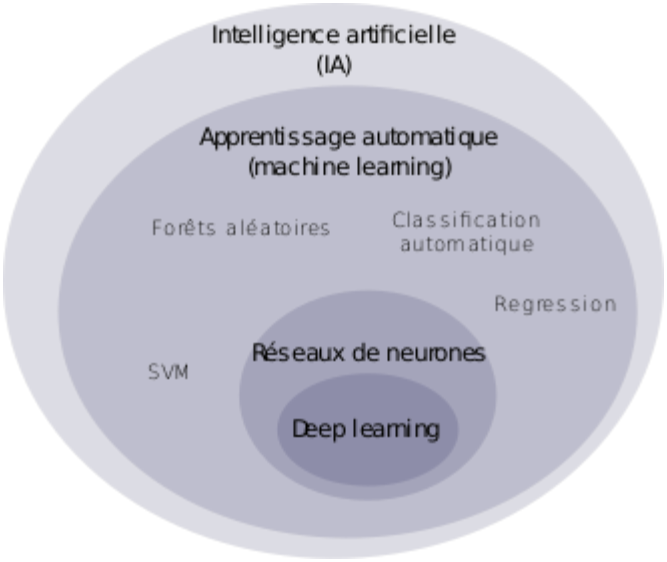


Figure 11: Schéma de l'intelligence artificielle**[13]**.

3-Domaines d'application de l'Intelligence artificielle:

L'IA est un domaine très vaste qui a de nombreuses applications. Comme première classification, on peut en distinguer trois types **[14]**:

- Des programmes qui imitent des capacités cognitives et reproduisent un raisonnement humain, par exemple un diagnostic médical, la configuration d'une central téléphonique, la planification d'une mission spatiale, ou la recherche de régularités dans une grande base de données.

- Des programmes qui imitent des capacités sensorielles et sont capables de reconnaître des formes ou des objets, ou bien de comprendre la parole en langue naturelle.

- Des programmes qui imitent des capacités sensorielles et sont capables de réagir de façon autonome à leur environnement, par exemple des robots ou agents autonomes.

4-Approches de l'intelligence artificielle:

La cybernétique est un terme proposé en 1947 par le mathématicien américain Norbert Wiener pour désigner une vision unifiée de l'automatique, de l'électronique et de la théorie de l'information de Shannon. Son étymologie renvoie à la notion de pilote, ou de gouverneur,

exprimant ainsi son objectif d'étude de l'information et de sa structure dans les interactions entre systèmes.

Le paradigme cybernétique est basé sur quelques concepts clefs très simples :

- La boîte noire, qui est un système dont « on décide » d'ignorer le fonctionnement interne mais dont on connaît la fonction d'association entre entrée et sortie
- L'émetteur, qui envoie une information vers son environnement
- Le récepteur, qui intègre des informations depuis son environnement
- L'émission, la transmission, et la réception de l'information comme flux
- La rétroaction comme information de retour, indispensable à toute logique

  d'autorégulation Appliquée à la cognition, cette approche très globale de la dynamique de l'information va se décomposer en deux branches principales complémentaires l'une de l'autre :

- Le cognitivisme, qui assimile les processus mentaux à des manipulations symboliques à partir de règles, dans une approche top-down de la cognition
- Le connexionnisme, qui envisage la cognition comme un processus émergeant de réseaux d'unités simples interconnectées (le plus souvent des modélisations de réseaux neuronaux), dans une approche bottom-up**[15].**

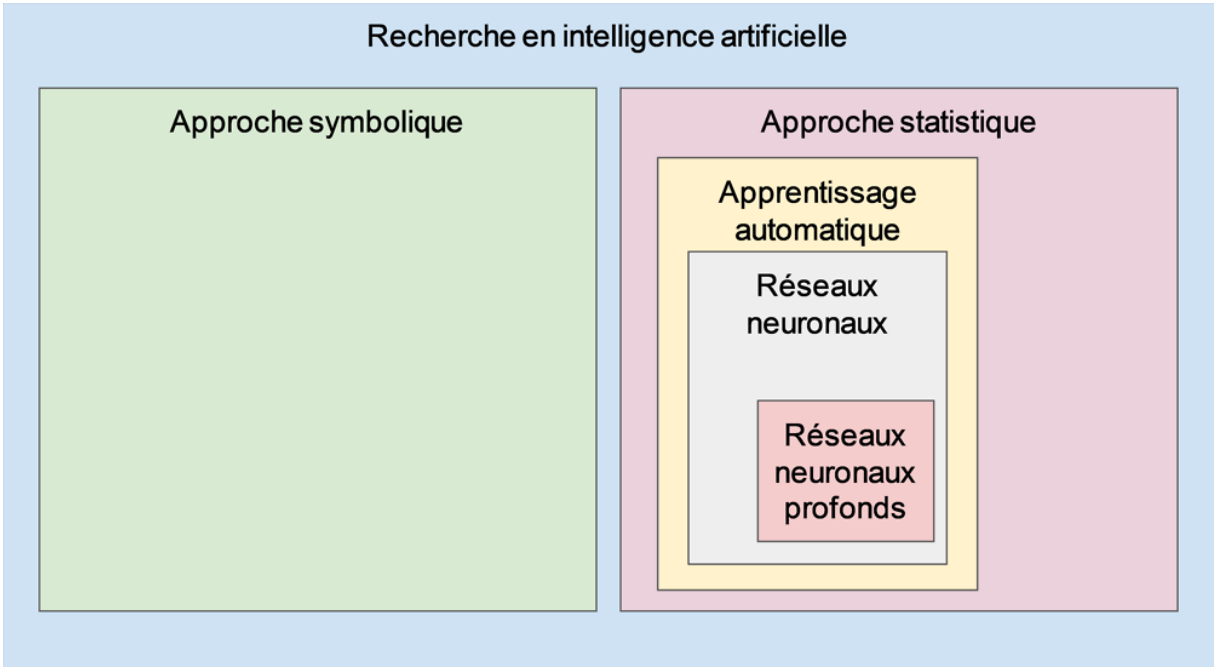


Figure N°12 : Schéma d'une approche de l'intelligence artificielle**[15].**

4-1-Apprentissage automatique (Machine Learning):

Derrière l'intelligence artificielle, se cache une multitude d'appareils qui ont un dénominateur commun : le machine learning. Cette technologie permet de stocker une grande quantité de données dans un cerveau ou réseau neuronal virtuel. On distingue l'intelligence artificielle forte de l'intelligence artificielle faible. La première inclut les machines capables d'agir de façon intelligente, mais aussi d'assimiler des concepts abstraits et d'avoir une véritable conscience proche des sentiments éprouvés par les êtres humains. Les machines qui se limitent à résoudre des problèmes entrent dans la catégorie d'intelligence artificielle faible **[16].**

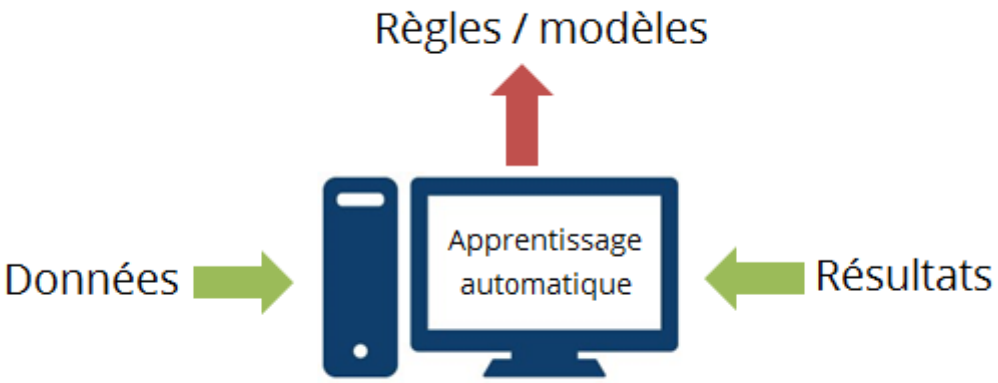


Figure 13 : Apprentissage automatique**[16].**

4-2-Réseaux de neurones:

Les réseaux de neurones figurent parmi les techniques les plus répondues dans le domaine de l'intelligence artificielle, pour cela ils sont utilisés dans plusieurs programmes et logiciels de commande des réseaux électriques; dans des domaines d'application variés: la stabilité, la sécurité, le diagnostic des défauts, la protection, le dispatching économique, l'analyse des harmoniques, l'écoulement des puissances, la prédiction de la demande électrique, la prédiction des pannes ,...,etc**[17].**

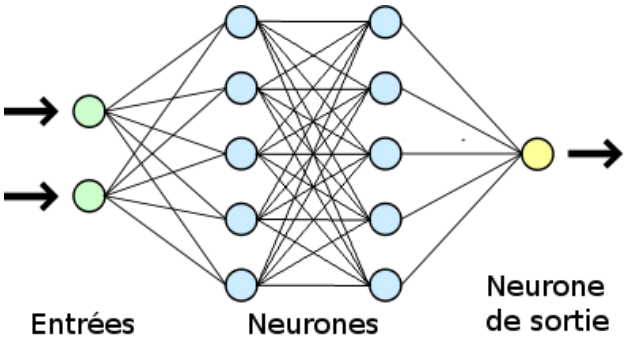


Figure 14 : Réseau de neurones**[17].**

4-3-Calcul évolutif (méta-heuristiques):

En informatique, le calcul évolutif est une famille d'algorithmes d'optimisation globale inspirés de l'évolution biologique et du sous-domaine de l'intelligence artificielle et de l'informatique douce qui étudient ces algorithmes. En termes techniques, il s'agit d'une famille d'algorithmes qui résolvent des problèmes par essais et erreurs et qui ont un caractère d'optimisation métaheuristique ou stochastique.

Dans le calcul évolutif, un ensemble initial de solutions candidates est généré et mis à jour de manière itérative. Chaque nouvelle génération est produite en supprimant de manière stochastique les solutions les moins souhaitées et en introduisant de petits changements aléatoires. Dans la terminologie biologique, une population de solutions est soumise à la sélection naturelle (ou sélection artificielle) et à la mutation . En conséquence, la population va progressivement évoluer pour augmenter en condition physique**[18].**

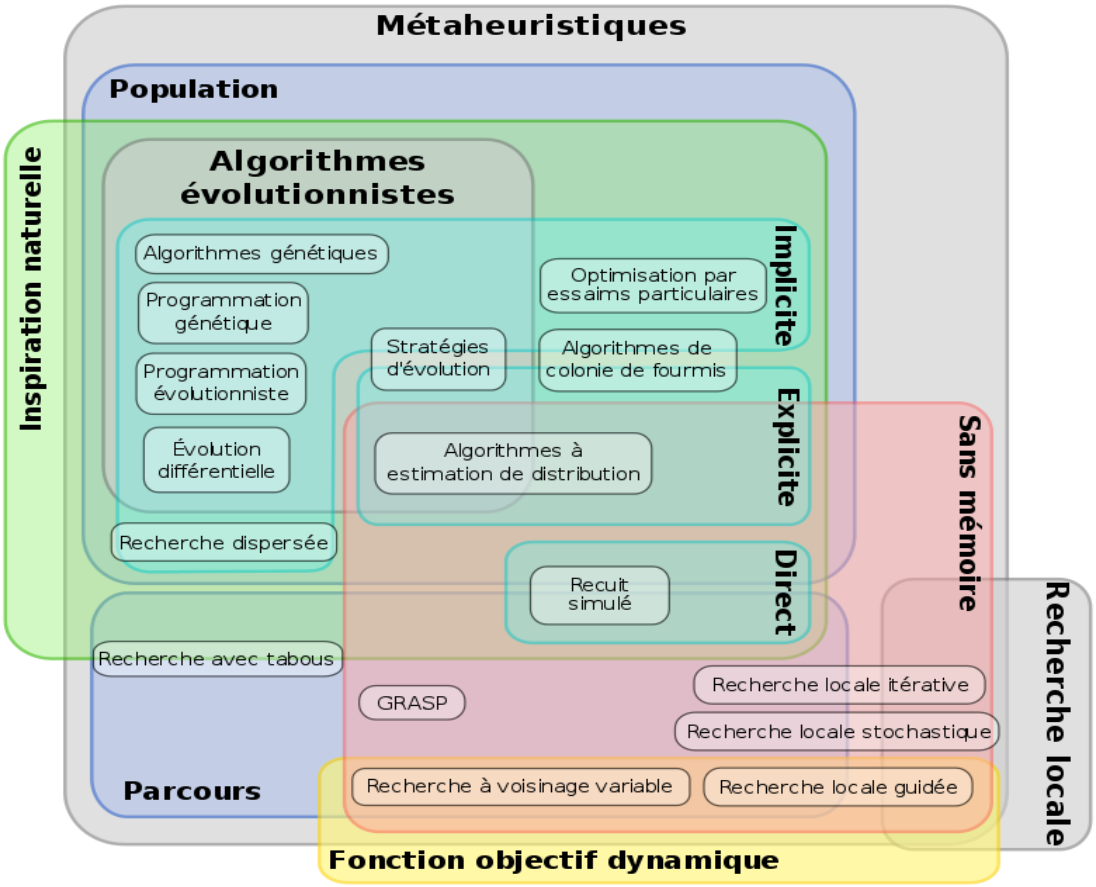


Figure 15 : Classification des méta-heuristiques**[18].**

4-4-Logique floue:

La logique floue a été introduite pour approcher le raisonnement humain à l'aide d'une représentation adéquate des connaissances. Son intérêt réside dans sa capacité à traiter l'imprécis, l'incertain et le vague. Elle est issue de la capacité de l'homme à décider et agir de façon pertinente malgré le flou des connaissances disponibles. Cependant, un système flou est difficile à appréhender. Sa commande et son réglage peuvent être relativement long. Il s'agit parfois beaucoup plus de tâtonnement que d'une réelle réflexion. Il manquait donc à la logique floue un moyen d'apprentissage performant pour régler un système flou, c'est les réseaux de neurones**[19].**

4-5-Systèmes experts:

Les systèmes expèrts, comme nouveaux outils informatiques, sont apparus, en France, au début des années 80. Leur caractéristique était alors d'être des objets techniques issus d'un champ particulier du traitement de l'information: l'Intelligence Artificielle (I.A.). Cette

filiation, porteuse d'une approche distincte en informatique, a conduit bien souvent à les considérer comme la réalisation des intentions de cette discipline. De fait, les perspectives initiales de l'I.A. étaient d'étendre le traitement de l'information à la quasi-totalité des activités humaines et de concevoir des outils pouvant simuler l'acte de penser. Ces visées radicales, affichées par les pères de l'I.A. aux États-Unis, dans les années 50, ont suscité des réactions critiques, tout d'abord de la part de philosophes, relayés bientôt par des chercheurs en I.A. même**[20].**

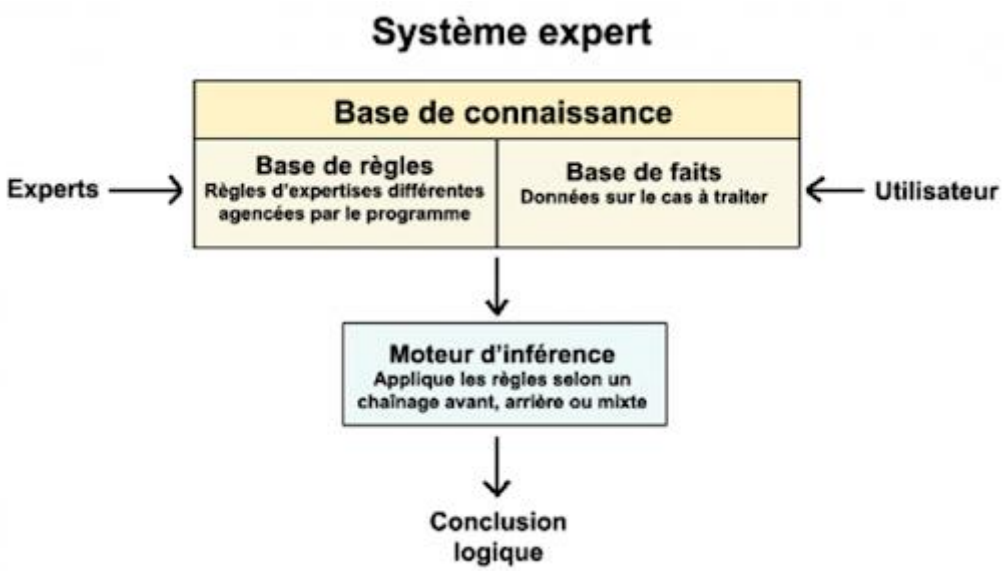


Figure 16 : Schéma d'un système expert**[20].**

5-Quelques applications des méthodes de l'intelligence artificielle dans l'alignement de séquences:

En utilisant l'optimisation des colonies de fourmis et des algorithmes génétiques, une nouvelle méthode d'alignement de séquences d'acides nucléiques par paires dans une séquence d'acide nucléique. La méthode **[21]** est largement un algorithme hybride utilisant l'optimisation des colonies de fourmis (ACO) et l'algorithme génétique simple. La méthode utilise d'abord l'ACO pour obtenir un ensemble d'alignements, qui sont ensuite traités par un algorithme génétique d'obtention, qui utilise des primitives sélections et un nouvel opérateur de mutation croisée multipoint pour générer des alignements précis.

RBT-GA **[22]** est une nouvelle métaheuristique pour la résolution du problème de l'alignement de séquences multiples. Pour résoudre le problème MSA en utilisant une méthodologie hybride de la technique de l'élastique et la métaheuristique (Algorithme génétique) (RBT-GA). Chaque réponse alignée est modélisée comme un chromosome

constitué de plusieurs Colonnes dans le cadre RBT. Ces électrodes sont similaires aux emplacements dans la séquence d'entrée susceptibles d'être apparentés et/ou biologiquement apparentés. Basé sur l'AG Le processus d'optimisation améliore progressivement ces chromosomes résultant en un ensemble de la plupart d'entre eux.

CHAPITRE 3 : Utilisation et évaluation des algorithmes génétiques pour l'alignement multiple de séquences

CHAPITRE 3 : Utilisation et évaluation des algorithmes génétiques pour l'alignement multiple de séquences

1-Introduction

Dans ce chapitre, nous allons présenter notre travail qui consiste à étudier les performances d'une méthode de l'intelligence artificielle qui est une métaheuristique sur le problème de l'alignement multiple de séquences. L'approche globale consiste à utiliser une mesure de qualité d'alignement multiple (une fonction objectif) et à l'optimiser à l'aide d'un algorithme génétique. Dans notre travail nous avons choisi l'Algorithme Génétique (AG), une métaheuristique qui a prouvé sont efficacité dans plusieurs domaines et qui est considéré parmi les méthodes approchées les plus efficaces pour traiter le problème de l'alignement multiple de séquences. Nous allons utiliser l'algorithme GA **[24]** et l'adapter pour notre étude et comparer ses performances avec un algorithme d'alignement multiple de séquences très répondu appelé ClustalW **[25]** dont le code source **[26]** est présenté dans l'annexe 1.

Ce chapitre est organisé comme suit : Dans une première partie nous allons présenter quelques définitions et notations liées à notre étude. Puis nous allons décrire comment nous une solution donnée et l'espace de recherche sont définis, et comment la fonction objectif est formulée. Ensuite nous allons présenter le scenario d'évaluation expérimentale ainsi que la présentation et la discussion des résultats. Nous clôturons ce chapitre par une conclusion.

2-Présentation de l'algorithme génétique :

Un algorithme génétique (AG) est un algorithme de recherche heuristique utilisé pour résoudre des problèmes de recherche et d'optimisation. Cet algorithme est un sous-ensemble d'algorithmes évolutionnaires, qui sont utilisés dans le calcul. Les algorithmes génétiques utilisent le concept de génétique et de sélection naturelle pour apporter des solutions aux problèmes.

À chaque étape, l'algorithme génétique utilise la population actuelle pour créer les enfants qui composent la prochaine génération. L'algorithme sélectionne un groupe d'individus dans la population actuelle, appelés parents, qui apportent leurs gènes (les entrées de leurs vecteurs) à leurs enfants. Le code source de cet algorithme **[24]** est présenté dans l'annexe 2.

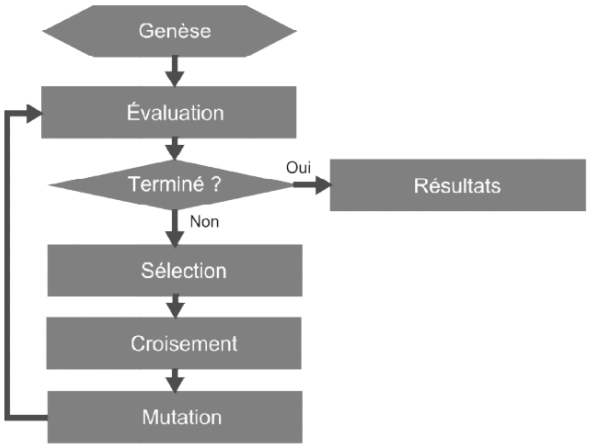


Figure 17 : Algorithme génétique**[24].**

Fonction objectif : L'évaluation des alignements est effectuée à l'aide d'une fonction qui est simplement une mesure de la qualité des alignements multiples. Le principe est de donner un coût à chaque couple de résidus alignés dans chaque colonne de l'alignement, et un autre coût aux gaps. Ceux-ci sont ajoutés pour donner le coût global de l'alignement. De plus, chaque paire de séquences se voit attribuer un poids lié à leur similarité avec les autres paires.

Individu : Chaque individu est représenté par un ensemble se séquences alignées. Le nombre de ces séquences est égal au nombre de séquences à alignées (T).

Population : est un ensemble d'individus. La population à une taille à définir N.

Meilleur individu : est l'individu (alignement) qui présente le meilleur score.

Sélection : Sélectionner les meilleurs individus (50% de la population)

Croisement et Mutation : Pour créer un enfant, un opérateur est sélectionné qui peut être un croisement (mélanger le contenu des deux parents) ou une mutation (modifier un seul parent). Chaque opérateur a une probabilité d'être choisi.

3-Scenarios d'évaluation et performance mesurée :

D'abord l'algorithme génétique considéré nommé GA-AMS est implémenté en langage Python. Afin de comparer les performances de cet algorithme, un autre algorithme appelé ClustalW a été considéré qui est également programmé en python pour résoudre le problème de l'alignement multiple de séquences.

L'algorithme GA-AMS est testé par rapport à ClustalW sur plusieurs instances de séquences. Chaque instance (ensemble de séquences à aligner) est générer aléatoirement par

un programme que nous avons développé en Python. Où nous avons varié le nombre de séquences N de chaque instance et les leurs taille T. Les informations sur les différentes instances d'évaluation sont indiquées sur le tableau 3.

Tableau 3 : Les différentes instances générées des séquences à aligner.

Instances	**Nombre de séquences (N)**	**Taille des séquences (T)** (Aléatoirement entre T/2 et T)
1	3	30
2	3	30
3	3	30
4	3	30
5	6	60
6	6	60
7	6	60
8	6	60
9	9	90
10	9	90
11	9	90
12	9	90
13	12	120
14	12	120
15	12	120
16	12	120

Les programmes GA-AMS et ClustalW sont exécutés pour mesurer les scores de l'alignement multiple pour chaque instance, sachant que nous avons utilisé la matrice PAM250 le calcul des scores avec une pénalité de -2 des gaps.

4-Paramètres expérimentaux :

Les valeurs des paramètres expérimentaux utilisées dans les différentes expériences

d'évaluation des algorithmes GA-AMS et ClustalW sont indiquées dans le Tableau 4.

Tableau 4 : Valeurs des paramètres expérimentaux.

Paramètre (unité)	**Valeur**
Longueur de la séquence (Bases)	30 , 60 , 90 , 120
Nombre de séquences	3 , 6 , 9 , 12
Nombre de d'itérations pour GA-AMS	1000

Les résultats obtenus pour les deux algorithmes sur les différentes instances sont indiqués dans le tableau 5.

Tableau 5 : Résultats des expériences.

		SCORE	
Taille des séquences (N)	**Nombre de séquences (N)**	**GA-AMS**	**ClustalW**
30	3	-94	-105
30	6	-752	-716
30	9	-3325	-3838
30	12	-9934	-9985
60	3	-183	-105
60	6	-2112	-2080
60	9	-6731	-7294
60	12	-17001	-17395
90	3	-204	-66

90	6	-3425	-2955
90	9	-11202	-9394
90	12	-30014	-32833
120	3	-179	-202
120	6	-4351	-4495
120	9	-13187	-13211
120	12	-35733	-37072

Pour une meilleure présentation de ces résultats, les figures de 18 à 21 fournissent une interprétation plus facile.

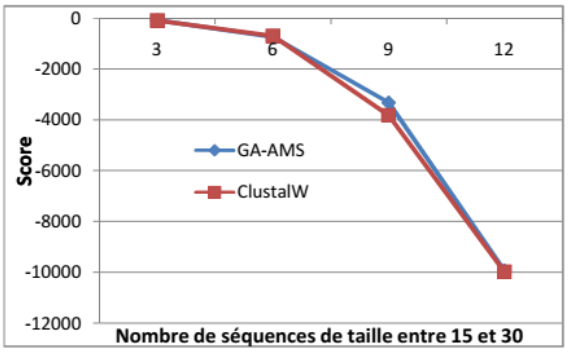


Figure 18 : Impact du nombre de séquence sur le score pour 15 et 30.

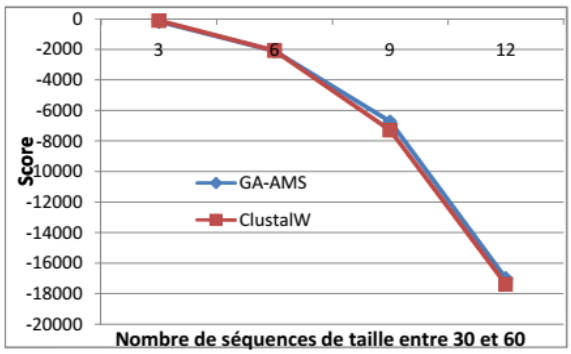


Figure 19 : Impact du nombre de séquence sur le score pour 30 et 60.

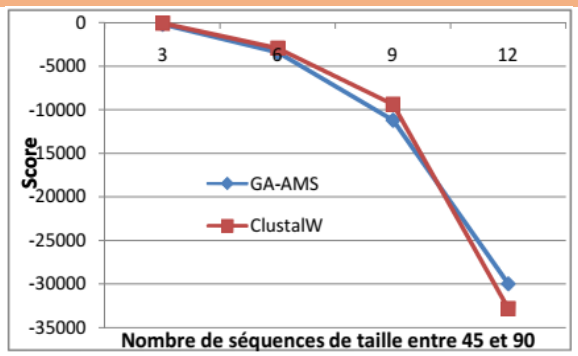


Figure 20 : Impact du nombre de séquence sur le score pour 45 et 90.

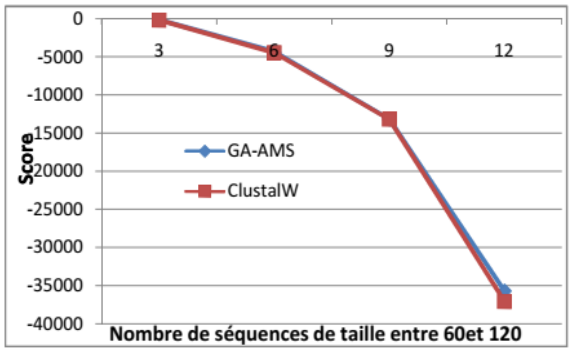


Figure 21 : Impact du nombre de séquence sur le score pour 60 et 120.

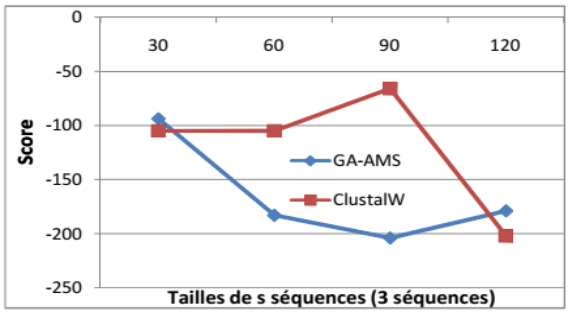


Figure 22 : Impact du nombre de séquence sur le score pour 3 séquences.

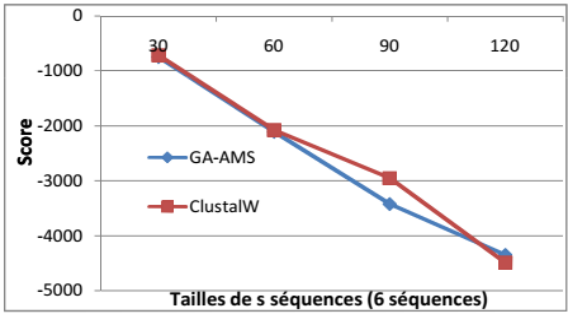


Figure 23 : Impact du nombre de séquence sur le score pour 6 séquences.

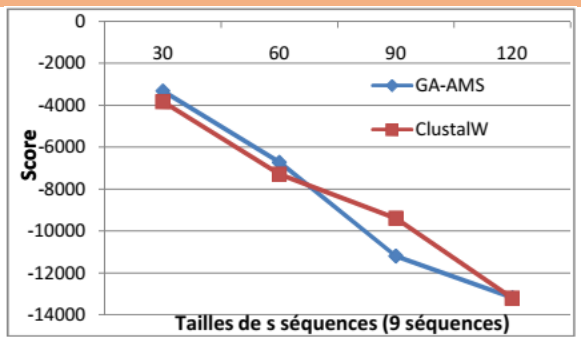


Figure 24 : Impact du nombre de séquence sur le score pour 9 séquences.

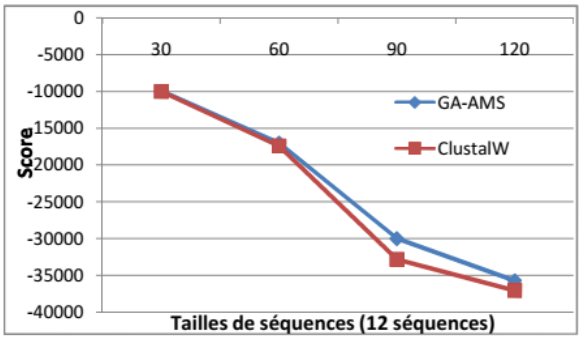


Figure 25 : Impact de la taille des séquences sur le score pour 12 séquences.

5-Interprétation et discussion des résultats :

Les figures 18 à 21montre une comparaison des deux algorithmes GA-AMS et ClustalW qui représente le score de l'alignement multiple de séquences en fonction du nombre de séquences à aligner. Comme nous pouvons le constater, les algorithmes GA-AMS et ClustalW montrent une dégradation du score de l'alignement multiple en augmentant le nombre de séquences à aligner. De même, Les figures22 à 25montre une comparaison des deux algorithmes GA-AMS et ClustalW qui représente le score de l'alignement multiple de séquences en fonction de la taille des séquences à aligner. Comme nous pouvons le constater, les algorithmes GA-AMS et ClustalW montrent une diminution du score de l'alignement multiple en augmentant la taille des séquences à aligner. Les résultats montrent que l'algorithme GA-AMS a de meilleur score d'alignement par rapport à ClustalW (11/16 point de tests). Malgré que ClustalW est un algorithme dédié pour le problème de l'alignement multiple de séquences, une simple méthode qui l'algorithme GA-AMS basée sur la métaheuristique AG montre des résultats intéressants expliqués par la capacité des techniques de l'intelligence artificielle, notamment les métaheuristiques, pour résoudre des problèmes difficiles.

6-Conclusion

Dans ce chapitre nous avons étudié la performance d'un algorithme approché, qui appartient au domaine de l'intelligence artificielle, qui utilise la métaheuristique GA pour traiter le problème de l'alignement multiples de séquences. Nous avons comparé ces performances avec un autre algorithme dédié pour traiter le même problème appelé ClustalW sur plusieurs scénarios d'évaluation. Les résultats ont montré que l'algorithme de la métaheuristique GA concurrence ClustalW et le surpasse dans plusieurs tests notamment dans les cas où le nombre de séquences est grand et les tailles des séquences sont longues.

Conclusion et Perspective

Conclusion et Perspective

Le travail présenté dans ce modeste mémoire vise à étudier et à montrer l'efficacité de l'intelligence artificielle notamment les métaheuristiques pour traiter le problème de l'alignement multiples de séquences. Nous nous sommes tout d'abord intéressés à présenter les notions de base sur l'alignement de séquences ainsi que les méthodes utilisées dans ce contexte. Nous avons ensuite présenté des notions de base des méthodes de l'intelligence artificielle. Vu que les méthodes algorithmiques les plus répondues présentent une structure plus complexe et sont généralement plus efficaces pour certains cas et moins efficaces pour d'autres, notre travail vient de montrer comment l'implication des méthodes de l'intelligence artificielle offre aussi de bons résultats. Dans ce sens, nous avons utilisé l'algorithme génétique pour traiter le problème de l'alignement de séquences. Nous avons comparé la performance de cet algorithme en termes de score avec un autre algorithme appelé ClustalW. Les résultats ont montré une efficacité acceptable de l'algorithme génétique par rapport à ClustalW sur les différents scénarios étudiés. Nous estimons que l'utilisation des méthodes de l'intelligence artificielle pour la résolution du problème l'alignement de séquences est un bon choix. L'étude expérimentale réalisée nous a permis de montrer l'efficacité de l'approche utilisée et c'est dans cette voie que nous souhaitons que plusieurs autres approches de l'intelligence artificielle seront implémentées et testées pour le problème de l'alignement de séquences.

Références Bibliographiques

Références Bibliographiques

[1] PascalHella. [En ligne] https://www.techno-science.net/glossaire-definition/Biologie-moleculaire.htm .

[2] Peter Clote, Feng Lou, and Alain Denise. A new approach to suboptimal pairwise sequence alignement. IASTED conference ComBio . Cambridge,UK : s.n., July 11-13,2011.

[3] Notes de cours de Bio-informatique. Pr DJEKOUN A. Pr HAMIDECHI M. A, P11.

[4] Nadira Benlahrache, (2007), Optimisation Multi-Objectif Pour l'Alignement Multiple de Séquences, Mémoire Présenté en vue de l'obtention du diplôme de Magistère en Informatique, Université Mentouri de Constantine, P18, 30.

[5] McDaniel, B. A. M., Grundy, F. J., Artsimovitch, I., Henkin, T. M. Transcription termination control of the S box system : direct measurement of S-adenosylmethionine by the leader RNA. Proc. Natl Acad. Sci. USA 100, 30833088, 2003.

Winkler, W. C., Nahvi, A., Sudarsan, N., Barrick, J. E., Breaker, RR. An mRNA structure that controls gene expression by binding S-adenosylmethionine. Nature Struct.Biol. 10, 701707, 2003 126

Sudarsan, N., Barrick, J. E., Breaker, RR. Metabolite Binding RNA domains are present in the genes of eukaryotes. RNA 9, 644647, 2003.(alignement de séquences)

[6] Céline Brochier-Armanet,(2003), Bioinformatique: alignement de séquences, Université Claude Bernard, Lyon 1, Laboratoire de Biométrie et Biologie évolutive (UMR 5558),Celine.brochier-armanet@univ-lyon1.fr, P9, 12.

[7] Jacques van Helden, Matrices de substitution, Jacques van Helden Jacques.van-Helden@univ-amu.fr Université d'Aix-Marseille, France Lab. Technological Advances for Genomics and Clinics

[8] Hess et al. BMC Bioinformatics (2016) 17:189, Addressing inaccuracies in BLOSUM computation improves homology search, performance, journal DOI 10.1186/s12859-016-1060-3.

Références Bibliographiques

[9]http://biochimej.univ angers.fr/Page2/BIOINFORMATIQUE/7ModuleBioInfoJMGE/99Matrice/1Matrice.htm , (consulté 12.50h, 15/08/2021).

[10] Céline Brochier-Armanet,(2003), Bioinformatique: alignement de séquences, Université Claude Bernard, Lyon 1, Laboratoire de Biométrie et Biologie évolutive (UMR 5558), Celine.brochier-armanet@univ-lyon1.fr, P 24.

[11] SERIBLI Houssam Eddine, (2015), Développement et Implémentation d'un Solveur Bio-inspiré pour l'Alignement de Séquences Biologiques, mémoire de fin d'étude Présenté pour l'obtention du diplôme de MASTER, UNIVERSITE DE M'SILA, P 24-27.

[12] Génet, Pondération des gaps, Université de TOURS, Document modifié, le 13 septembre, 2007.

[13] https://journals.openedition.org/activites/4941#tocto1n, (Consulté le 01/08/2021, à 08.00h).

[14] https://www.futura-sciences.com/tech/definitions/informatique-intelligence-artificielle-555/,(Consulté le 19/08/2021, à 17.00h).

[15]https://books.google.dz/books?id=mh4SHxyGecC&lpg=PR5&ots=Z9XeRxn8ha&dq=domaine%20d'application%20de%20l'intelligence%20artificiell, e&lr&hl=ar&pg=PA10#v=onepage&q=domaine%20d'application%20de%20 l'intelligence%20artificielle&f=false (Consulté le 28/08/2021,à 14.00h).

[16]https://www.quantmetry.com/blog/approches-de-lintelligence-artificielle-partie-i/ (Consultée 29/08/2021, à 15.00h).

[17] https://www.futura-sciences.com/tech/definitions/informatique-intelligence-artificielle-555/(Consulté le 06/07/2021, à 08.30h).

[18] M. DRAIDI ABDELLAH, (2010), Répartition économique de l'énergie électrique utilisant les techniques d'intelligence artificielle, Université Mentouri de Constantine, p28.

[19] https://www.wigi.wiki/wiki/fi/Evolutionary_computing, (Consulté le 17/07/2021, à 11.00h).

Références Bibliographiques

[20] Azeddine Chaiba , (2010),Commande de la machine asynchrone a double alimentation par des techniques de L'intelligence Artificielle , Thése Présentée à l'Université de Batna En vue de l'obtention du diplôme de doctorat en science se électrotechnique, p 85.

[21] Martine BLANC, Elsie CHARRON, Michel FREYSSENET, (1989), le «développement» des systèmes-experts en entreprise, GROUPEMENT D'INTÉRÊT PUBLIC « MUTATIONS INDUSTRIELLES » CNRS, Entreprises, Agences Publiques, Ministères Groupement de Recherche CNRS numéro 43 0002 26, boulevard Richard Lenoir - 75011 PARIS, p14.

[22] S.R. Jangam a , N. Chakraborti, (2006), A novel method for alignment of two nucleic acid sequences ,using ant colony optimization and genetic algorithms, a Department of Biotechnology and Biochemical Engineering, Indian Institute of Technology, Kharagpur 721302, India, p8.

[23] Javid Taheri and Albert.Zomaya , (2008), RBT-GA: A novel metaheuristic for solving the multiple sequence alignment problem technical report 628, Université Sydné, p 06.

[24] https://github.com/AbdurrahimDerric/multiple_sequence_alignment_genetic_algorithm (Consulté le 01/09/2021, à 10.00h).

[25] https://github.com/ah-rasoulian/MultipleSequenceAlignment (Consulté le 01/09/2021, à 08.00h).

[26] https://github.com/ah-rasoulian/MultipleSequenceAlignment ah-rasoulian/ Multiple Sequence Alignment, (Consulté le 17/08/2021, à 16.00h).

Annexes

Annexe 1 :Code source du GA-AMS.

```
import pickle
import random
from PAM import PAM250




def main():
    size = 20  #size of population
    gap_penalty = -2
    sequences, k = read_sequences('sequences.txt')
    population, seq_lengths = initialize(sequences, k, size)
    # print("pop size ", len(population))
    # print(population)
    genetic_algorithm(population)


def read_sequences(path):
   k = 0
   sequences = []
   with open(path,'r') as reader:
      for text in reader:
        text= text.replace('\n','')
        text = text.upper()
        sequences.append(text)
        k+=1
   return sequences,k


def initialize(sequences,k,size):
    longest = 0
    seq_lengths = []
    for seq in sequences:
        if len(seq) > longest:
            longest = len(seq)
        seq_lengths.append(len(seq))
    print('longest ', longest)
    # print(seq_lengths)
    individual = []
    population = []
    for i in range(size):
        j= 0;
        for seq in sequences:
            gaps = longest - seq_lengths[j] #get the lenght of this sequence
            front_gaps = random.randint(0,gaps)
            end_gaps = gaps - front_gaps
            seq = ('-'*front_gaps) + seq + ('-' * end_gaps)
            #print(seq)
```

```
            j+=1
            individual.append(seq)
        #print('\n')
        population.append(individual)
        individual= []
    return population,seq_lengths

def sum_of_pairs(population,gap_penalty):
    dummy  = 1
    total_score = 0
    sequence_number = 0;
    score_array = []
    for sequences in population:
        for seq_1 in sequences:
            sequence_number+=1
            score = 0
            dummy = 0
            for seq_2 in sequences:
                if seq_1 is seq_2 or dummy < sequence_number:
                    dummy+=1
                else:
                    #print(seq_1, '    ', seq_2)
                    i= 0
                    i = int(i)
                    for letter_i  in seq_1:
                        try:
                            if(letter_i == '-' or seq_2[i] == '-'):
                                score+=gap_penalty;
                            else:
                                score +=PAM250[letter_i][seq_2[i]]
                            i+=1
                        except:
                            pass
                    # print(score)
                    total_score+= score
        # print('\n')
        dummy = 1
        sequence_number = 0
        score_array.append(total_score)
        total_score = 0;

    return score_array

def selection(population,sum,pr):  #select a individual according to its possibility
    p = 0
    p = random.randint(1, sum)
    #    print(p)
    c = 0
    while pr[c] < p:
        c += 1
```

```
    return (population[c])


def halves_of_populations(population,score_array):
    scored_population = []
    new_population = []
    index = int(0)
    for indiv in population:
        scored_population.append([indiv, score_array[index]])
        index += 1
    scored_population = sorted(scored_population, key=lambda a_entry: a_entry[1])
    scored_population.reverse()
    half_of_population = scored_population[:len(scored_population) // 2]
    other_half = scored_population[len(scored_population) // 2:]

    sum = 0
    pr = []
    # print("other half is ",other_half)
    for i in other_half:
        if(i[1] < 0):
            space = 1
        else:
            space = i[1]
        sum += space + 1
        pr.append(sum);
    return half_of_population,other_half,sum,pr


def check_region(individual,gap_penalty,threshold,region_threshold):
    default_start = len(individual[0])//4
    default_end = len(individual[0])//2
    start_index = 0
    end_index = 0
    region = False
    previous_start = default_start
    previous_end = default_end
    region_length = 0

    first_seq = individual[0]
    letter_index = 0
    for letter_1 in first_seq:
      score =0
      for seq in individual:
          if first_seq == seq:
              pass
          else:
              letter_2 = seq[letter_index]
              if (letter_1 == '-' or letter_2 == '-'):
                  score += gap_penalty;
              else:
```

```
                score += PAM250[letter_1][letter_2]

        if(score > threshold):
            # print('the score is', score)
            if(region == True):
                end_index+=1
            else:
                # print('started')
                region = True
                start_index = int(letter_index)
                end_index = int(letter_index) +1
        elif(region == True) and (end_index - start_index  > region_length):
            #print('bigger')
            previous_start = start_index
            previous_end = end_index -1
            region_length = end_index - start_index
            region = False
        else:
            region = False
        letter_index += 1;

    if region_length > region_threshold:
        sequence = individual[0]
        sequence = sequence[previous_start:previous_end]
        return previous_start,previous_end
    else:
        return default_start,default_end




def reproduce(individual1,gap_penalty,threshold,region_threshold):
    start_index,end_index =
check_region(individual1,gap_penalty,threshold,region_threshold)
    child = []
    temp = []

    for seq in individual1:
        region = seq[start_index:end_index]
        temp.append(region)

    index = 0
    head = ''
    tail = ''

    for seq in individual1:
        head = seq[:start_index]
        tail = seq[end_index:]
        head = swap_gaps(head)
        tail =swap_gaps(tail)
```

```
        seq = head + temp[index] + tail
        child.append(seq)
        index+=1;
    return child


def swap_gaps(seq):
    new_seq = ''
    seq = list(seq)
    index =0
    while index < len(seq):
        if seq[index] == '-':
            swap_direction = random.randint(0, 1)
            if swap_direction == 1 and index < len(seq)-1:
                temp = seq[index+1]
                seq[index+1] = seq[index]
                seq[index] = temp
            elif index >0:
                temp = seq[index - 1]
                seq[index- 1] = seq[index]
                seq[index] = temp

        index+=1

    for letter in seq:
        new_seq+= letter
    return new_seq

def best_individual(half_of_population):
    indiv = half_of_population[0]

    return indiv


def print_indiv(indiv):
    score  = indiv[1]
    indiv = indiv[0]
    for seq in indiv:
        print(seq)
    print('the score: ', score)
def best_among_best(indivs):
    score = 0
    best = indivs[0]

    for indiv in indivs:
        if indiv[1] >score:
            best = indiv
            score = indiv[1]
    return best
```

```
def genetic_algorithm(population):
    found = False
    iterarion = 0
    size = 20
    best_individuals = []
    gap_penalty = -2
    threshold = 3
    region_threshold = 1
    print("pop size ", len(population))
    print(population)


    while (iterarion < 1000):
        iterarion += 1
        #if iterarion % 100 == 2:
            #print("Generation Count : " + str(iterarion) + '\n')

        new_population = []
        score_array = sum_of_pairs(population, gap_penalty)
        half_of_population, other_half,sum,pr =
halves_of_populations(population,score_array)
        best_indiv = best_individual(half_of_population)
        #print_indiv(best_indiv)
        best_individuals.append(best_indiv)

        for x in half_of_population:
            child = reproduce(x[0],gap_penalty,threshold,region_threshold)
            new_population.append(child)

        for i in range(size//2):
            x = selection(other_half,sum,pr)
            y = selection(other_half,sum,pr)
            child = reproduce(x[0],gap_penalty,threshold,region_threshold)
            if child not in new_population:
                new_population.append(child)
            else:
                #print('//////////////////////////exist')
                i -= 1

        population = new_population

    found = best_among_best(best_individuals)
    print("\n\n the best alignment")
    print_indiv(found)
    print("\n\n\n")

main()
```

Annexe 2 : Code source du ClustalW.

```
import math
from collections import Counter
from PAM import PAM250


class GuideTreeNode:
    def __init__(self, node_id, sequence_number, priority, child_1=None, child_2=None,
child_1_distance=None,
                 child_2_distance=None):
        self.id = node_id
        self.sequence_number = sequence_number
        self.priority = priority
        self.child_1 = child_1
        self.child_2 = child_2
        self.child_1_distance = child_1_distance
        self.child_2_distance = child_2_distance


def print_tree(input_tree):
    for key in input_tree.keys():
        print(vars(input_tree.get(key)))


def delete_children(parent_id):
    global node_list
    if parent_id is None:
        return
    else:
        node_list.remove(parent_id)
        delete_children(tree.get(parent_id).child_1)
        delete_children(tree.get(parent_id).child_2)


def find_multiple_sequence_alignment(parent_node):
    parent_node: GuideTreeNode
    if parent_node.child_1 is None:
        return [parent_node.sequence_number]
    else:
        left_sequences = find_multiple_sequence_alignment(tree.get(parent_node.child_1))
        right_sequences =
find_multiple_sequence_alignment(tree.get(parent_node.child_2))

        left_consensus_sequence = find_consensus_sequence(left_sequences)
        right_consensus_sequence = find_consensus_sequence(right_sequences)

        align_left, align_right, distance = global_align(left_consensus_sequence,
right_consensus_sequence)
```

```
        align_based_on_consensus_alignment(left_sequences, left_consensus_sequence,
align_left)
        align_based_on_consensus_alignment(right_sequences, right_consensus_sequence,
align_right)

        left_sequences.extend(right_sequences)
        return left_sequences


def find_consensus_sequence(sequences_index):
    sequences_index: []
    consensus_sequence = ""
    for i in range(len(sequences[sequences_index[0]])):
        ith_chars = []
        for seq_id in sequences_index:
            ith_chars.append(sequences[seq_id][i])
        consensus_sequence += Counter(ith_chars).most_common(1)[0][0]
    return consensus_sequence


def align_based_on_consensus_alignment(sequences_index, consensus_sequence,
aligned_consensus):
    global sequences
    for i in range(len(aligned_consensus)):
        if aligned_consensus[i] == '-':
            if len(consensus_sequence) < i + 1:
                consensus_sequence += '-'
                for seq_id in sequences_index:
                    sequences[seq_id] += '-'
            elif consensus_sequence[i] != '-':
                consensus_sequence = consensus_sequence[0:i] + '-' +
consensus_sequence[i:len(consensus_sequence)]
                for seq_id in sequences_index:
                    sequences[seq_id] = sequences[seq_id][0:i] + '-' +
sequences[seq_id][i:len(sequences[seq_id])]
    return sequences_index


def calculate_score00000(final_sequences):
    double_gap_score = -2
    gap_score = -2
    match_score = 1
    mismatch_score = -1
    score = 0
    for i in range(len(final_sequences[0])):
        ith_chars = []
        for seq in final_sequences:
            ith_chars.append(seq[i])
        for first in range(len(ith_chars)):
```

```
            for second in range(first + 1, len(ith_chars)):
                if ith_chars[first] == '-' and ith_chars[second] == '-':
                    score += double_gap_score
                elif ith_chars[first] == '-' or ith_chars[second] == '-':
                    score += gap_score
                elif ith_chars[first] == ith_chars[second]:
                    score += match_score
                else:
                    score += mismatch_score
    return score

def calculate_score(sequences,gap_penalty):
    dummy  = 1
    total_score = 0
    sequence_number = 0;
    score_array = []

    for seq_1 in sequences:
        sequence_number+=1
        score = 0
        dummy = 0
        for seq_2 in sequences:
            if seq_1 is seq_2 or dummy < sequence_number:
                dummy+=1
            else:
                #print(seq_1, '    ', seq_2)
                i= 0
                i = int(i)
                for letter_i  in seq_1:
                    try:
                        if(letter_i == '-' or seq_2[i] == '-'):
                            score+=gap_penalty;
                        else:
                            score +=PAM250[letter_i][seq_2[i]]
                        i+=1
                    except:
                        pass
                # print(score)
                total_score+= score
    # print('\n')
    return total_score


def global_align(x, y, s_match=1, s_mismatch=-1, s_gap=-2):
    A = []
    for i in range(len(y) + 1):
        A.append([0] * (len(x) + 1))
    for i in range(len(y) + 1):
        A[i][0] = s_gap * i
    for i in range(len(x) + 1):
```

```
        A[0][i] = s_gap * i
    for i in range(1, len(y) + 1):
        for j in range(1, len(x) + 1):
            A[i][j] = max(
                A[i][j - 1] + s_gap,
                A[i - 1][j] + s_gap,
                A[i - 1][j - 1] + (s_match if (y[i - 1] == x[j - 1] and y[i - 1] != '-') else 0) + (
                    s_mismatch if (y[i - 1] != x[j - 1] and y[i - 1] != '-' and x[j - 1] != '-') else 0) + (
                    s_gap if (y[i - 1] == '-' or x[j - 1] == '-') else 0)
            )
    align_X = ""
    align_Y = ""
    i = len(x)
    j = len(y)

    while i > 0 or j > 0:
        current_score = A[j][i]

        if i > 0 and j > 0 and (
                ((x[i - 1] == y[j - 1] and y[j - 1] != '-') and current_score == A[j - 1][i - 1] +
s_match) or
                ((y[j - 1] != x[i - 1] and y[j - 1] != '-' and x[i - 1] != '-') and current_score == A[j
- 1][
                    i - 1] + s_mismatch) or
                ((y[j - 1] == '-' or x[i - 1] == '-') and current_score == A[j - 1][i - 1] + s_gap)
        ):
            align_X = x[i - 1] + align_X
            align_Y = y[j - 1] + align_Y
            i = i - 1
            j = j - 1
        elif i > 0 and (current_score == A[j][i - 1] + s_gap):
            align_X = x[i - 1] + align_X
            align_Y = "-" + align_Y
            i = i - 1
        else:
            align_X = "-" + align_X
            align_Y = y[j - 1] + align_Y
            j = j - 1
    return align_X, align_Y, A[len(y)][len(x)]


def create_guide_tree(distance_matrix, N):
    global tree
    while N > 2:
        net_divergence_r = {}
        # calculating r values
        for seq1_id in distance_matrix.keys():
            _sum = 0
            for seq2_id in distance_matrix.get(seq1_id).keys():
                _sum += distance_matrix[seq1_id][seq2_id]
```

```
            net_divergence_r[seq1_id] = _sum

        # calculating new distance matrix
        distance_prime_matrix = {}
        for seq1_id in distance_matrix.keys():
            distance_column = {}
            for seq2_id in distance_matrix.get(seq1_id).keys():
                distance_column[seq2_id] = distance_matrix[seq1_id][seq2_id] - (
                        net_divergence_r.get(seq1_id) + net_divergence_r.get(seq2_id)) / (N - 2)
            distance_prime_matrix[seq1_id] = distance_column

        # find minimum distance index
        minimum_distance_index1 = 0
        minimum_distance_index2 = 0
        minimum_distance_value = math.inf
        for seq1_id in distance_prime_matrix.keys():
            for seq2_id in distance_prime_matrix.get(seq1_id).keys():
                if distance_prime_matrix[seq1_id][seq2_id] < minimum_distance_value or
(distance_prime_matrix[seq1_id][seq2_id] == minimum_distance_value and
(min(tree.get(seq1_id).priority, tree.get(seq2_id).priority) <
min(tree.get(minimum_distance_index1).priority,
tree.get(minimum_distance_index2).priority))):
                    minimum_distance_index1, minimum_distance_index2 = seq1_id, seq2_id
                    minimum_distance_value = distance_prime_matrix[seq1_id][seq2_id]

        # create parent node of 2 minimum distanced nodes
        child_1_distance =
distance_matrix[minimum_distance_index1][minimum_distance_index2] / 2 + (
                net_divergence_r[minimum_distance_index1] -
net_divergence_r[minimum_distance_index2]) / (
                                2 * (N - 2))
        child_2_distance =
distance_matrix[minimum_distance_index1][minimum_distance_index2] -
child_1_distance

        new_node = GuideTreeNode(len(tree), None,
min(tree.get(minimum_distance_index1).priority,
                                         tree.get(minimum_distance_index2).priority),
                           minimum_distance_index1, minimum_distance_index2,
child_1_distance,
                           child_2_distance)
        tree[new_node.id] = new_node

        # create new distance matrix
        new_distance_matrix = {}
        new_node_column = {}
        for seq1_id in distance_matrix.keys():
            distance_column = {}
            if seq1_id not in [minimum_distance_index1, minimum_distance_index2]:
                for seq2_id in distance_matrix[seq1_id].keys():
```

```
                if seq2_id not in [minimum_distance_index1, minimum_distance_index2]:
                    distance_column[seq2_id] = distance_matrix[seq1_id][seq2_id]
            new_node_distance = (distance_matrix[minimum_distance_index1][seq1_id] +
                                 distance_matrix[minimum_distance_index2][seq1_id] -

distance_matrix[minimum_distance_index1][minimum_distance_index2]) / 2
            distance_column[new_node.id] = new_node_distance
            new_node_column[seq1_id] = new_node_distance
            new_distance_matrix[seq1_id] = distance_column
        new_distance_matrix[new_node.id] = new_node_column
        distance_matrix = new_distance_matrix

        N -= 1

def read_sequences(path):
  k = 0
  sequences = []
  with open(path,'r') as reader:
     for text in reader:
       text= text.replace('\n','')
       text = text.upper()
       sequences.append(text)
       k+=1
  return sequences,k

def main():
   gap_penalty = -2  #dms
   global tree, node_list, sequences
   #n = int(input())  # getting counts of sequences

   sequences, n = read_sequences('sequences.txt')


   for i in range(n):
      #new_sequence = input().upper()  # getting sequences that we are going to align
      #sequences.append(new_sequence)
      tree[i] = GuideTreeNode(len(tree), i, i)

   distance_matrix = {}

   for seq1_id in tree.keys():
      distance_column = {}
      for seq2_id in tree.keys():
         if seq1_id != seq2_id:
            align_x, align_y, distance =
global_align(sequences[tree.get(seq1_id).sequence_number],
                                          sequences[tree.get(seq2_id).sequence_number])
            distance_column[seq2_id] = distance
      distance_matrix[seq1_id] = distance_column
```

```
    # creating guide tree
    create_guide_tree(distance_matrix, n)

    # delete first child of root and all its descendants to find second child
    node_list = list(tree.keys())
    rootChild1ID = tree.get(len(tree) - 1).id
    delete_children(rootChild1ID)
    # getting second child of root node
    rootChild2ID = node_list[len(node_list) - 1]

    # create root node and add it to the tree
    root = GuideTreeNode(len(tree), None, min(tree.get(rootChild1ID).priority,
tree.get(rootChild2ID).priority),
                         rootChild1ID, rootChild2ID)
    tree[root.id] = root

    find_multiple_sequence_alignment(root)

    # printing MSA
    for sequence in sequences:
        print(sequence)
    # printing MSA score
    print(calculate_score(sequences,gap_penalty))


tree = {}
node_list = []
sequences = []

if __name__ == '__main__':
    main()
```

Annexe 3 : Le code source en python du programme qui généré les différents distances de séquences à aligner.

```
import math
from collections import Counter
import random
import string

def generate_sequences(n:int, l:int) -> list:
    # N is the number of sequences to create
    # l is the length of the sequences
    DNA = []
    base_arr =
["A","C","D","E","F","G","H","I","K","L","M","N","P","Q","R","S","T","V"]
    Midle=int(math.floor(l/2))
    for t_index in range(0, n):
        DNA.append("")
        len2=Midle+int(math.floor(random.random() * Midle))
        for _ in range(0, len2):
            r = int(math.floor(random.random() * 18))
            DNA[t_index] = DNA[t_index] + base_arr[r]
    return DNA

L=90
N=12


file1 = open("sequences_"+"N_"+str(N)+"_L_"+str(L) +".txt","w")
DNA= generate_sequences(N,L)
#print (DNA)
for t_index in range(0, N):
    file1.write(DNA[t_index]+"\n")

file1.close()value="Retour"/></form></p>
```

Résumés

Résumé

En bioinformatique, un alignement de séquence est un moyen d'organiser les séquences d'ADN, d'ARN ou de protéines pour identifier des régions de similitude qui peuvent être une conséquence de relations fonctionnelles, structurelles ou évolutives entre les séquences. Les séquences alignées de résidus nucléotidiques ou d'acides aminés sont généralement représentées sous forme de lignes dans une matrice. Des espaces sont insérés entre les résidus afin que des caractères identiques ou similaires soient alignés dans des colonnes successives. Les méthodes de l'intelligence artificielle semblent un moyen prometteur dans ce type de problème vu la grande quantité d'information à traiter et vu l'incapacité des méthodes classiques à fournir des solutions en un temps opportun. Notre travail est une tentative d'investiguer quelques méthodes de l'intelligence artificielle au profit de l'alignement de séquences.

Mots clés : Alignement de séquences ; Intelligence artificielle ; Algorithme génétique ; Bioinformatique.

Abstract

Abstract

In bioinformatics, sequence alignment is a way to organize DNA, RNA, or protein sequences to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between sequences. Aligned sequences of nucleotide or amino acid residues are generally shown as lines in a matrix. Spaces are inserted between the residues so that identical or similar characters are aligned in successive columns. Artificial intelligence methods seem a promising medium in this type of problem given the large amount of information to be processed and the inability of traditional methods to provide solutions in a timely manner. Our work is an attempt to investigate some artificial intelligence methods for the benefit of sequence alignment.

Key words: Alignment of sequences; Artificial intelligence ; Genetic algorithm; Bioinformatics.

ملخص

في المعلوماتية الحيوية ، تعد محاذاة التسلسل طريقة لتنظيم تسلسلات الحمض النووي أو الحمض النووي الريبي أو البروتين لتحديد مناطق التشابه التي قد تكون نتيجة للعلاقات الوظيفية أو الهيكلية أو التطورية بين التسلسلات. تظهر المتواليات المحاذية لبقايا النوكليوتيدات أو الأحماض الأمينية بشكل عام كخطوط في مصفوفة. يتم إدراج المسافات بين المخلفات بحيث تتم محاذاة الأحرف المتطابقة أو المتشابهة في أعمدة متتالية. تبدو أساليب الذكاء الاصطناعي وسيلة واعدة في هذا النوع من المشاكل نظرًا للكم الكبير من المعلومات التي يجب معالجتها وعدم قدرة الأساليب التقليدية على توفير الحلول في الوقت المناسب. عملنا هو محاولة للتحقيق في بعض أساليب الذكاء الاصطناعي لصالح محاذاة التسلسل.

الكلمات المفتاحية : محاذاة التسلسلات ذكاء اصطناعي ؛ الخوارزمية الجينية المعلوماتية الحيوية.

Année universitaire : 2020/2021 **Présenté par : BENOTMANE Meriem**
CHEIKH BOUKAL Selma

Implication des méthodes de l'intelligence artificielle dans l'alignement des séquences

Mémoire de fin de cycle pour l'obtention du diplôme de Master en Bioinformatique.

Résumé

En bioinformatique, un alignement de séquence est un moyen d'organiser les séquences d'ADN, d'ARN ou de protéines pour identifier des régions de similitude qui peuvent être une conséquence de relations fonctionnelles, structurelles ou évolutives entre les séquences. Les séquences alignées de résidus nucléotidiques ou d'acides aminés sont généralement représentées sous forme de lignes dans une matrice. Des espaces sont insérés entre les résidus afin que des caractères identiques ou similaires soient alignés dans des colonnes successives. Les méthodes de l'intelligence artificielle semblent un moyen prometteur dans ce type de problème vu la grande quantité d'information à traiter et vu l'incapacité des méthodes classiques à fournir des solutions en un temps opportun. Notre travail est une tentative d'investiguer quelques méthodes de l'intelligence artificielle au profit de l'alignement de séquences.

Mots clés : Alignement de séquences ; Intelligence artificielle ; Algorithme génétique ; Bioinformatique.

Jury d'évaluation :

- **Président de jury : Dr.BELLIL Ines** **MCA.UFM. Constantine 1.**
- **Encadreur : Dr.DAAS Mohamed Skander** **MCA. UFM. Constantine 1.**
- **Examinateur : Dr.DJOUDI Brahim** **MCB. UFM. Constantine 1.**

Date de soutenance : 23/09/2021