REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DES MATHEMATIQUE ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

N° :……..

DOMAINE : MATHEMATIQUE ET INFORMATIQUE

FILIERE : INFORMATIQUE

OPTION : INFORMATIQUE DECISIONNELLE ET OPTIMISATION

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par: DJERBOUAI Khalil

Intitulé

Alignement multiple des séquences protéiques par l'algorithme de recherche tabou

Soutenu devant le jury composé de :

Dr. LAMICHE Chaabane	Université de M'sila	Président
Dr. YAGOUBI Rached	Université de M'sila	Rapporteur
Dr. AMRI Said	Université de M'sila	Examinateur

Année universitaire : 2017 /2018

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DES MATHEMATIQUE ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

N° :……..

DOMAINE : MATHEMATIQUE ET INFORMATIQUE

FILIERE : INFORMATIQUE

OPTION : INFORMATIQUE DECISIONNELLE ET OPTIMISATION

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par: DJERBOUAI Khalil

Intitulé

Alignement multiple des séquences protéiques par l'algorithme de recherche tabou

Soutenu devant le jury composé de :

Dr. LAMICHE Chaabane	Université de M'sila	Président
Dr. YAGOUBI Rached	Université de M'sila	Rapporteur
Dr. AMRI Said	Université de M'sila	Examinateur

Année universitaire : 2017 /2018

DÉDICACES

Nous voila à la fin d'un parcours c'est long mais plein de bons événements, de meilleurs souvenirs et d'inoubliables amitiés tissées tout au long de nos années d'études ;

Ce mémoire n'est pas seulement la finalité d'une année d'étude mais le résultat de tant d'années de travail et de recherche du savoir ;

Cet humble travail que nous allons présenter ci-après, n'a pu avoir lieu si tout ceux qui me sont chers ne se pas consacrés et venus à mon aide ;

Je dédier mon modeste travail à :

Ma famille mon père et ma mère qui ne m'a jamais laissé sentir de manque et qui ont pris soin de moi depuis ma naissance, mes frères et sœurs et tous mes amis.

C'est pour cela que je leur dédie mon travail en leur disant :

« ***Je vous aime et merci beaucoup pour tout ce que vous m'avez offert*** »

Khalil

REMERCIEMENTS

Tout d'abord nous remercions notre bon Dieu le tout puissant, pour son aide et pour nous avoir donné le courage et la patience afin d'accomplir notre travail dans les meilleurs conditions.

Nous tenons aussi à remercier notre promoteur

Mr YAGOUBI Rached pour son suivi et ses précieux conseils durant l'évolution de ce travail.

nos remerciements qui vont également :

A tous nos enseignants qui ont contribué à notre formation, ainsi que tous les personnels de Collège.

A toute notre promotion pour tous les bons moments qu'on a passés ensemble.

Khalil

Table des matière

Introduction général

Chapitre I : Introduction à la bioinformatique

1. Introduction ……..……..……..……..……..……..……..……..……..……..……..……..5
2. Notion de base de biologie moléculaire ……..……..……..……..……..……..……..……..5
   2.1. Cellule ……..……..……..……..……..……..……..……..……..……..……..……..5
   2.2. ADN (Acide DésoxyriboNucléique)……..……..……..……..……..……..……..……6
      2.2.1. Nucléotides…….....6
      2.2.2. Séquence d'ADN…..……..……..……..……..……..……..……..……..……..……..……..……7
      2.2.3. Structure …..……..……..……..……..……..……..……..……..……..……..……..……..……..8
   2.3. L'ARN (Acide RiboNucléique) …..……..……..……..……..……..……..……..……..……..……..……..8
   2.4. Protéine …..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……8
      2.4.1. Protéines structurelles …..……..……..……..……..……..……..……..……..……..……..……..……..8
      2.4.2. Enzymes …..……..……..……..……..……..……..……..……..……..……..……..……..……..……..8
      2.4.3. Transporteuses …..……..……..……..……..……..……..……..……..……..……..……..……..……..8
3. Définition et thèmes en bioinformatique …..……..……..……..……..……..……..……..……..……..……..9
   3.1. Alignement de séquences …..……..……..……..……..……..……..……..……..……..……..……..……..9
   3.2. Phylogénie…..……..……..……..……..……..……..……..……..……..……..……..……..……..……..10
   3.3. Recherche de motifs …..……..……..……..……..……..……..……..……..……..……..……..……..11
   3.4. Prédiction de structures…..……..……..……..……..……..……..……..……..……..……..……..……..12
4. Représentation informatique …..……..……..……..……..……..……..……..……..……..……..……..……..12
   4.1. Séquence…..……..……..……..……..……..……..……..……..……..……..……..……..……..……..12
   4.2. Alphabet …..……..……..……..……..……..……..……..……..……..……..……..……..……..……..12
   4.3. Sous séquence…..……..……..……..……..……..……..……..……..……..……..……..……..……..13
   4.4. Longueur …..……..……..……..……..……..……..……..……..……..……..……..……..……..……..13
5. Format de séquence …..……..……..……..……..……..……..……..……..……..……..……..……..……..13
   5.1. Fasta format …..……..……..……..……..……..……..……..……..……..……..……..……..……..……..13
   5.2. MSF …..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……..13
   5.3. Clustal…..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……14
6. Les Banques des données Biologique …..……..……..……..……..……..……..……..……..……..……..……14
   6.1. Les Banques des Séquences Nucléiques …..……..……..……..……..……..……..……..……..……..……15
   6.2. Les Banques des Séquences Protéiques …..……..……..……..……..……..……..……..……..……..16
   6.3. Les Banques de Motif …..……..……..……..……..……..……..……..……..……..……..……..……..16
7. Conclusion…..……..……..……..……..……..……..……..……..……..……..……..……..……..……..……17

Chapitre II : Alignement par paires

1. Introduction…..……..……..……..……..……..……..……..……..……..……..……..……..……..……19
2. Définitions…..……..……..……..……..……..……..……..……..……..……..……..……..……..……19
   2.1. Définition 1..……..……..……..……..……..……..……..……..……..……..……..……..……..……19
   2.2. Définition 2..……..……..……..……..……..……..……..……..……..……..……..……..……..……19
3. Évaluation d'un Alignement …..……..……..……..……..……..……..……..……..……..……..……..……20

4. Distance et similarité entre deux séquences…….21
4.1. Similarité et homologie……..21
4.2. La distance de Hamming…….21
4.3. Le Système de Score……..22
4.4. Les Matrices de Substitution……..22
4.4.1. Matrices de Scores pour l'ADN……22
4.4.2. Matrices de score pour les protéines……...23
4.5. Evaluation des brèches……..25
4.5.1. Les brèches à cout constant……26
4.5.2. Les brèches à cout affine……..26
5. Principe de la programmation dynamique……..27
5.1. Problème et sous-problème……27
5.2. Principe d'optimalité……27
5.3. Programmation dynamique……27
6. Les Méthodes d'Alignement par paires……..28
6.1. Alignement Global……28
6.2. Alignement Local……..29
7. Comparaison avec les Banques de Séquences……30
8. Les algorithmes (Fasta et Blast) ……..30
8.1. Fasta (Fast alignement) ……30
8.2. Blast (Basic Local Alignment Search Tool) ……30
9. Conclusion……..31

Chapitre III : Alignement multiple des séquences

1. Introduction……33
2. Définition……33
3. Les utilisation en bioinformatique……..33
4. Évaluation d'un alignement multiple de séquence ……34
5. Les fonctions d'évaluation……..34
5.1. La somme des paires……34
5.2. La somme des paires pondérées……35
5.3. La fonction Coffee……36
6. Les approches d'Alignements Multiple de Séquences ……..38
6.1. L'Approche Exacte……38
6.2. L'Approche Itérative……39
6.3. L'Approche Progressive……..39
7. Classification des Méthodes……. 40
7.1. L'approche exacts……40
7.1.1. La méthode MSA……..41
7.1.2. Méthode basé sur la programmation dynamique……42
7.1.3. La méthode DCA……43
7.2. L'approche progressif ……..44
7.2.1. Clustal……..45
7.2.2. MUSCLE...……..46

7.3. L'approche itérative…..……..……..……..……..……..……..……..……..……47
7.3.1. SAGA…..……..……..……..……..……..……..……..……..……..……47
7.3.2. Autre méthode…..……..……..……..……..……..……..……..……..……48
8. Les benchmarks…..……..……..……..……..……..……..……..……..……..……..48
9. Conclusion…..……..……..……..……..……..……..……..……..……..……..…49

Chapitre IV : l'algorithme recherche tabou pour l'alignement multiple de séquence

1. Introduction……..……..……..……..……..……..……..……..……..……..……51
2. La recherche tabou…..……..……..……..……..……..……..……..……..……..…51
3. Réalisation et expérimentation…..……..……..……..……..……..……..……..……54
3.1. Environnement de développement…..……..……..……..……..……..……..……..54
3.1.1. Environnement matériel…..……..……..……..……..……..……..……..……..54
3.1.2. Environnement logiciel …..……..……..……..……..……..……..……..……..55
3.2. Présentation de l'algorithme…..……..……..……..……..……..……..……..……..56
3.2.1. Génération de la solution initiale …..……..……..……..……..……..……..57
3.2.2. La fonction d'évaluation…..……..……..……..……..……..……..……..……..57
3.2.3. Structure de voisinage (Neighborhood) …..……..……..……..……..……..…57
3.2.4. La liste tabou…..……..……..……..……..……..……..……..……..……..…58
3.2.5. Critère d'aspiration…..……..……..……..……..……..……..……..……..……..59
3.2.6. Critères de résiliation…..……..……..……..……..……..……..……..……59
4. Les données de test…..……..……..……..……..……..……..……..……..……..…59
5. Critères d'évaluation…..……..……..……..……..……..……..……..……..……..…..60
6. Résultats obtenus par l'algorithme recherche tabou…..……..……..……..……60
6.1. Par la matrice PAM250 …..……..……..……..……..……..……..……60
6.2. Par la matrice BLOSUM62…..……..……..……..……..……..……..……..……..…64
6.3. Analyse des résultat pour la recherche tabou…..……..……..……..……..……..68
7. Conclusion…..……..……..……..……..……..……..……..……..……..……..…71

Conclusion générale

Bibliographie

Liste des figure

Figure I.1 : *Schéma d'une cellule animale*

Figure I.2 : *Structure de l'ADN*

Figure I.3 : *Interrogation d'une base de données*

Figure II.1 : *Alignement de deux séquences protéiques [4]*

Figure II.2 : *Score d'un alignement [4]*

Figure IV.1 : *Environnement matérial*

Figure IV.2 : *(a) détection d'un gap et caractère, (b) le changement.*

Figure IV.3 : *Matrice binaire pour détecter le gap.*

Figure IV.4 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=100).*

Figure IV.5 : *Temps d'exécution pour (TL=20,nbr_itération=100).*

Figure IV.6 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=200).*

Figure IV.7 : *Temps d'exécution pour (TL=20,nbr_itération=200).*

Figure IV.8 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=500).*

Figure IV.9 : *Temps d'exécution pour (TL=20,nbr_itération=500).*

Figure IV.10 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=100).*

Figure IV.11 : *Temps d'exécution pour (TL=20,nbr_itération=100).*

Figure IV.12 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=200).*

Figure IV.13 : *Temps d'exécution pour (TL=20,nbr_itération=200).*

Figure IV.14 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=500).*

Figure IV.15 : *Temps d'exécution pour (TL=20,nbr_itération=500).*

Figure IV.16 : *Taux d'amélioration de score global pour chaque itération lorsque en utilisant la matrice PAM250 .*

Figure IV.17 : *Taux d'amélioration de score global pour chaque itération lorsque en utilisant la matrice BLOSUM62 .*

Figure IV.18 : *L'histogramme des moyen des Taux d'amélioration de score.*

Liste des tableaux

Tableaux I.1 : *tableaux de protéine*

Tableaux II.1 : *Matrice de Score Pour l'ADN* [12]

Tableaux II.2 : *Matrice de transition* [12]

Tableaux II.3 :*La matrice BLAST* [12]

Tableaux II.4 : *Matrice PAM 250* [13]

Tableaux II.5 : *Matrice BLOSUM 62* [13]

Tableaux III.1 : *Mémoire nécessaire pour un alignement multiple [7]*

Tableaux IV.1 : *Les données de test utilisées.*

Tableaux IV.2 : *Les résultat obtenus pour (TL=20,nbr_itération=100) .*

Tableaux IV.3 : *Les résultat obtenus pour (TL=20,nbr_itération=200) .*

Tableaux IV.4 : *Les résultat obtenus pour (TL=20,nbr_itération=500).*

Tableaux IV.5 : *Les résultat obtenus pour (TL=20,nbr_itération=100).*

Tableaux IV.6 : *Les résultat obtenus pour (TL=20,nbr_itération=200).*

Tableaux IV.7 : *Les résultat obtenus pour (TL=20,nbr_itération=500).*

Tableaux IV.8 : *Tout les résultat obtenus par (PAM250 et BLOSUM62).*

Introduction générale

Introduction générale

Afin de comprendre les mécanismes de fonctionnement du vivant, les biologistes ont besoin d'extraire des informations et des connaissances à partir des données biologiques, les interpréter et les analyser. Les données biologiques sont stockées dans des banques de données comme par exemple la banque des gènes (GenBank). Elle a été crée en 1982 et elle contenait 680338 bases de nucléotides dans 606 séquences. Actuellement, dans sa version 185 datée d'aout 2011, GenBank contient plus de 130 milliards de bases de nucléotides dans plus de 142 millions de séquences. Avec cette vitesse de croissance des banques de données biologiques et la grande disponibilité de ces données, l'utilisation de l'outil informatique est incontournable. La Bioinformatique se propose comme une science capable de fournir des moyens et des outils pour satisfaire les besoins des biologistes. La bioinformatique traite différents problèmes parmi lesquelles nous citons : la phylogénie, la recherche de motifs, la prédiction des structures et l'alignement de séquences. Ce dernier est très important dans la mesure ou il constitue un problème à part entière, mais il est également utilisé comme point de départ pour d'autres problèmes de bioinformatique. L'alignement de séquences permet de découvrir des similitudes biologiques entre les séquences (nucléiques ou protéiques) et de déterminer les correspondances entre résidus. Nous pouvons distinguer deux types de problèmes dans le domaine d'alignement de séquences selon le nombre de séquences traitées. Aligner deux séquences est appelé alignement par paires. Ce problème peut être résolu de manière exacte à l'aide de la programmation dynamique. Nous appelons alignement multiple tout alignement de plus de deux séquences. Ce problème a été démontré NP-complet et il ne peut être résolu par une méthode exacte que pour des séquences de petites tailles et dont le nombre est très réduit. Différents algorithmes existent pour l'alignement multiple de séquences. Nous pouvons

les classer selon trois catégories :

- Les algorithmes exacts sont minoritaires, ils ne peuvent être utilisés que pour des alignements ne comportant que peu de séquences car ils sont très gourmands en ressources CPU et mémoire.
- Les algorithmes progressifs qui consistent à aligner des sous-groupes de séquences. Ils commencent par réaliser des alignements de deux séquences, puis ces alignements sont à leur tour alignés entre eux. L'algorithme s'arrête une fois toutes les séquences regroupées. Ces méthodes sont simples, rapides et donnent généralement des alignements de bonnes qualités. Cependant, leur inconvénient majeur est la perte d'informations au cours du

processus d'alignement, car traiter des séquences deux à deux est moins précis que de les traiter toutes ensemble.

- Les algorithmes itératifs sont basés sur des méthodes plus variées que les algorithmes progressifs. Ils ont comme point commun de réaliser l'alignement de séquences en prenant en compte toutes les séquences simultanément. Le problème de ces algorithmes est qu'ils nécessitent en générale des temps de calculs tr`es importants.

Malgré l'existence d'un nombre très important d'algorithmes, aucun d'entre eux ne permet d'obtenir la solution optimale dans tous les cas. L'existence de benchmark, comme Balibase, permet une évaluation plus facile des nouveaux algorithmes.

Notre mémoire est organisé en quatre chapitres. Le premier contient une introduction à la bioinformatique, pour cela nous présentons quelques notions de biologie moléculaire et les différents thèmes de la bioinformatique. Le deuxième chapitre est consacré au cas particulier de l'alignement de deux séquences. Nous y exposons le problème ainsi que des algorithmes exacts pour le résoudre. Le troisième chapitre présente le problème d'alignement multiple de séquences proprement dit. Ce problème étant NP-Complet, nous exposons quelques uns des algorithmes permettant de résoudre ce problème. Le dernier chapitre contient la contribution personnelle. Nous détaillons tout d'abord l'algorithme recherche tabou ensuite nous présentons les résultats obtenu et l'évaluation de l'algorithme recherche tabou en avais une comparaison avec deux matrices de substitution, PAM250 et BLOSUM62 par l'algorithme recherche tabou. Le manuscrit se termine par une conclusion générale.

CHAPITRE I

Introduction à la bioinformatique

1. Introduction

Le terme de << bioinformatique >> date du début des années 80 . le concept sous-jacent de traitement de l'information biologique est bien plus vieux. Durant les années 60 , la biologie moléculaire a eu besoin de modélisation formelle, ce qui a mené à la création des <<biomathématique>> [1].

L'apparition de la bioinformatique n'est donc pas une conséquence de la génomique (séquençage d'un génome et son interprétation), mais plutôt une de ses fondations [1].

La bioinformatique est l'étude de l'information biologique . Ce n'est pas simplement l'application à la biologie de l'informatique ; c'est une branche à part entière de la biologie. La bioinformatique actuelle se concentre surtout sur l'étude des séquences d'ADN et sur le repliement des protéines [1].

2. Notion de base de biologie moléculaire

2.1. Cellule

La cellule est l'unité de base de tout organisme (une sorte de pièce de légo). Un corps humain est un immense assemblage, consistué de des ces pièces. La première distinction que l'on peut faire parmis les organismes vivants sépare ceux dont les cellules ont un noyau (les eucaryotes, comme nous), et ceux qui n'ont pas de noyau, plus primitifs (les procaryotes, comme les bactéries). Parmis les êtres dont les cellules ont un noyau, certains n'ont qu'une seule cellule (les protistes, comme les amibes), d'autres sont constitués des très nombreuses cellules [2].

Une cellule se présente comme une sorte de sac. Son enveloppe extérieure est appelée membrane plasmique. Cette membrane est un assemblage de lipides et de protéines (la bicouche lipidique ressemblant à celle d'une bulle de savon) . Ce sac contient une gelée appelée cytoplasme dans laquelle flottent de petits compartiments, eux aussi limités pas une membrane lipidique. Chacun de ces compartiments, ou org-anite [2]. Est spécialisé dans une tâche particulière :

- Le noyau : il contient l'ADN, c'est à dire toute l'information génétique [2].
- Les mitochondries et chloroplastes: ce sont les centrales énergétiques de la cellule[2].
- Le réticulum endoplasmique, l'appareil de golgi : nécessaires à la synthèse des protéines [2].
- Les lysosomes : ce sont les stations d'épuration de la cellule [2].
- Les vacuoles : remplies de lipide, elles contiennent les réserves énergétiques [2].

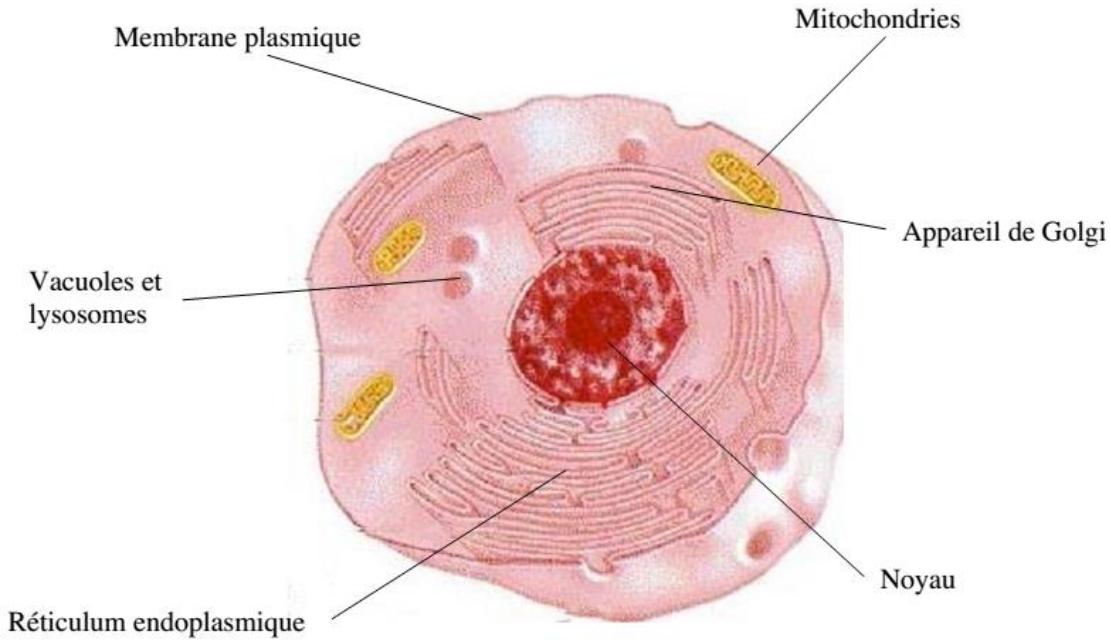

Figure I.1 : *Schéma d'une cellule animale*

2.2. ADN (Acide DésoxyriboNucléique)

L'ADN est un acide de la famille des acides nucléiques, composé d'une succession de nucléotides. On peut définir l'ADN d'un organisme comme l'ensemble des informations génétiques nécessaires à l'édification, au fonctionnement et à la reproduction de chaque organisme [7].

2.2.1. Nucléotides

Les nucléotides sont constitués à partir d'une base azotée (Adénine, Thymine, Guanine, Cystosine) d'un sucre et d'un groupement phosphate. Par convention on utilise la première lettre de chacune des bases pour appeler les nucléotides [7].

Les nucléotides peuvent être rangés en deux catégories. Les bases C et T sont dites pyrimidiques, et les bases A et G sont dites puriques. Lorsque l'on parle d'ADN, la représentation que l'on fait est souvent celle d'une structure en double hélice. Les deux brins constituant cette double hélice ne sont pas formés indépendamment. A chaque base d'un brin est associée sur l'autre brin une base complémentaire [7].

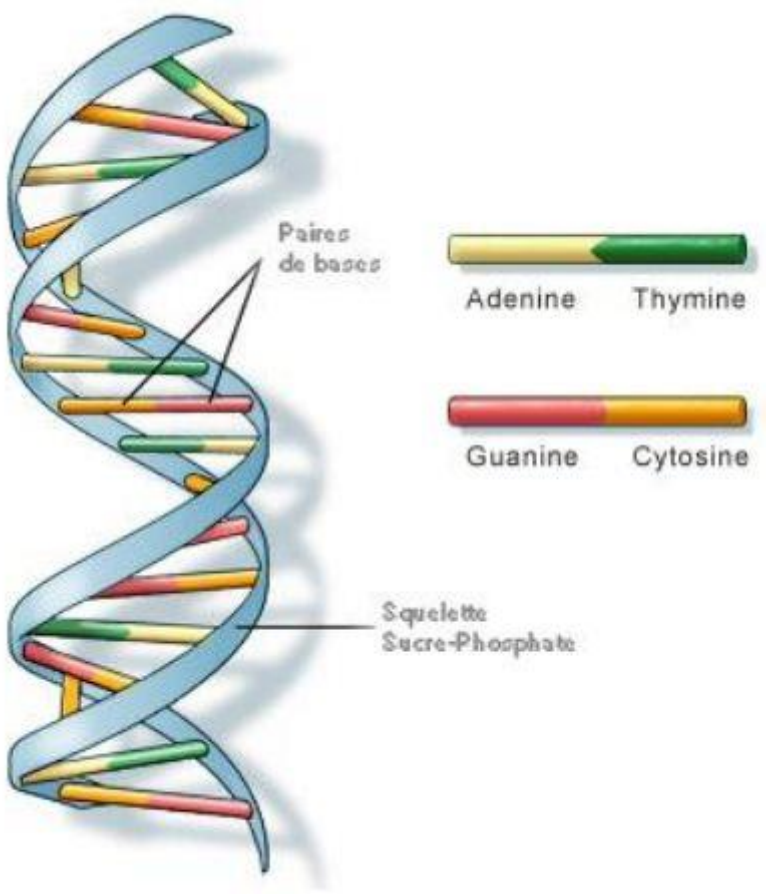

Figure I.2 : *Structure de l'ADN*

Cette complémentarité permet donc de définir totalement l'ADN à partir d'un seul des deux brins .

2.2.2. Séquence d'ADN

On appelle séquence d'ADN la lecture séquentielle de l'ensemble des bases constituant un brin d'ADN [7].

2.2.3. Structure

Cette représentation ordonnée des bases constitue la structure primaire de la séquence. La représentation de l'ADN sous forme de double hélice est appelée la structure secondaire [7].

2.3. L'ARN (Acide RiboNucléique)

L'ARN est obtenu à partir d'un des deux brins d'ADN [7]. A chaque nucléotide de la séquence d'ADN va correspondre le nucléotide complémentaire, sauf pour l'Adénine [7]. En effet, pour l'ARN, le complémentaire de l'Adénine n'est pas la Thymine mais l'Uracile [7].

- Transcription

La transcription est le mécanisme qui permet de dupliquer un brin $d'ADN$ sous forme $d'ARN$ [7].

$L'ARN$ est subdivisé en trois catégories, chacune ayant un rôle différent [7].

- $L'ARN$ de transfert ($ARNt$) est utilisé dans la phase de traduction de $l'ARN$ en protéines [7].
- $L'ARN$ ribosomique ($ARNr$) forme une trame sur les ribosomes, afin de permettre aux protéines synthétisées par les ribosomes de se fixer [7].
- $L'ARN$ messager ($ARNm$) est utilisé chez les eucaryotes pour véhiculer l'information génétique du noyau vers le cytoplasme (substance de la cellule qui entoure le noyau) [7].

2.4. Protéine

Les protéines sont les macromolécules les plus importantes. Elles sont responsables de presque de toutes les réactions biochimiques qui ont lieu à l'intérieur de la cellule. Les protéines sont de sortes différentes et avec une variété de fonctionnalités. Certaines d'entre elles incluent,[4].

2.4.1 : Protéines structurelles : elles sont les bases de construction des divers tissus [4].

2.4.2 : Enzymes : elles catalysent les réactions chimiques essentielles qui auraient pris beaucoup de temps pour se produire[4].

2.4.3 : Transporteuses : elles portent les éléments chimiques qui font partie de l'organisme à d'autres (par exemple les hémoglobines qui portent l'oxygène) [4].

Une séquence protéique est une collection ordonnée de lettres choisis dans l 'alphabet .

$= \{A,\ C,\ D,\ E,\ F,\ G,\ H,\ I\ ,\ K,\ L,\ M,\ N,\ P,\ Q,\ R,\ S,\ T,\ V,\ W,\ Y\}.$

Où chacune des lettres correspond à un acide aminé [4].

Nom	Code		Nom	Code
Alanine	A		Leucine	L
Arginine	R		Lysine	K
Asparagine	N		Méthionine	M
Aspartate	D		Phénylalanine	F
Cystéine	C		Proline	P
Glutamate	E		Sérine	S
Glutamine	Q		Thréonine	T
Glycine	G		Tryptophane	W
Histidine	H		Tyrosine	Y
Isoleucine	I		Valine	V

Tableaux I.1 : *tableaux de protéine*

La synthèse de protéine se produit dans des structures appelées les ribosomes situées dans la cellule mais endehors du noyau . Le modèle de la protéine est dans l'ADN, située dans le noyau. Donc il y a un besoin d'un " messager " pour transférer l'information à partir de l'ADN aux ribosomes. L'ARN est ce messager (ARNm). Il est synthétisé en utilisant l'ADN comme modèle. Ce processus s'appelle la transcription[4].

La structure de la protéine : La protéine possède quatre structures : primaire, secondaire, tertiaire et quartenaire. Parmi les paradigmes de la biologie moléculaire, celui de la relation entre structure et fonction. La structure secondaire ou tertiaire peut inférer la fonction d'une protéine. Donc connaître la structure va faciliter l'identification de sa fonction et par conséquent son importance pour tout l'organisme [16].

3. Définition et thèmes en bioinformatique

3.1. Alignement de séquences

L'alignement de séquences est une problématique importante de la bioinformatique. En effet aligner des séquences constitue un problème à part entière, mais il est également utilisé comme point de départ pour d'autres problèmes de bioinformatique [7]. Le problème de l'alignement de séquence sera détaillé dans le chapitre suivent :

3.2. Phylogénie

La phylogénie ou reconstruction phylogénétique peut être définie comme la reconstruction de l'histoire évolutionnaire d'un ensemble d'espèces. L'évolution entre les espèces est représentée sous forme d'un arbre (phylogénétique) dont les branches indiquent le degré de proximité entre les espèces [8].

Il existe deux façons de caractériser les espèces :

- soit en se basant sur leurs phénotypes, c'est à dire l'ensemble des caractéristiques qu'elles expriment (suivant leur apparence) [8].
- soit en se basant sur leurs génotypes, c'est à dire l'ensemble des caractéristiques comprisent dans leurs génomes et qu'elles peuvent éventuellement exprimer [8].

Le premier genre d'approche basé sur les phénotypes est utilisée lorsqu'on ne dispose d'aucune information sur les génomes des espèces. Depuis le lancement des différents programmes de séquençage des génomes on a de plus en plus tendance à utilisé le second type d'approche [8].

Ici nous ne nous focaliserons que sur le second type de caractérisation. Nous utiliserons des séquences nucléotidiques (ou d'acides aminés) pour distinguer les espèces entre elles [8].

Il existe 3 types d'approches pour s'attaquer au problème de reconstruction phylogénétique

- les approches basées sur les **distances** utilisent une mesure de distance ou de similarité afin de regrouper des séquences proches, en anglais on parle de clustering. Ces approches sont très efficaces car elles sont basées sur un algorithme de complexité polynomiale ce qui les rend particulièrement adaptées pour des études à grande échelle[8].
- les approches axées sur les **caractères** cherchent le meilleur arbre possible dans une topologie d'arbre étant donné un critère d'optimalité. Le critère le plus largement utilisé est le critère de maximum de parcimonie (Maximum Parsimony Criterion) pour lequel on considère que le meilleur arbre est celui qui requiert le minimum de changements. Le problème de maximum de parcimonie est un problème NP-dur dans le cas général, il est donc nécessaire de faire appel à des techniques heuristiques et d'optimisation afin de le traiter efficacement dans un temps raisonnable [8].

- enfin, les approches statistiques principalement basées sur la méthode de maximum de vraisemblance (Maximum Likelihood) et l'approche bayésienne [8].

Chacune de ces méthodes possède des avantages et des inconvénients. Faisant partie d'une équipe de recherche travaillant en optimisation combinatoire, nous avons choisi de nous intéresser à la recherche du Maximum de Parcimonie [8].

3.3. Recherche de motifs

Un motif (ou Pattern) au sens bioinformatique du terme , représente une expression qui permet de caractériser un ensemble de séquences d'ADN, d'ARN ou de protéines . Le motif peut concerner les structures primaires, secondaires et tertiaires. Le motif trouve notamment son intérêt dans la caractérisation des fonctions des protéines : si on était capable d'exhiber un motif pour chaque fonction alors on serait en mesure de prédire automatiquement la fonction associée à une protéine [7].

On distingue deux étapes dans la recherche de motif :

- la découverte qui, étant donné un ensemble de séquences, tente d'exhiber un motif commun à ces séquences. Il s'agit d'un problème complexe car on ne sait pas ce qui doit être trouvé. Dans le cas de séquences similaires, on peut utiliser un alignement multiple des séquences afin de trouver un motif simple [7].
- la recherche à proprement parler, qui concerne la détection d'un motif donné sur un ensemble de séquences. Ce problème est bien plus simple que le premier[7].

Les deux problèmes rencontrés dans la recherche de motifs concernent la définition du motif . Un motif généralement défini à partir d'un ensemble référence de séquences qui possèdent la même fonction:

- si le motif n'est pas assez fin, on risque de le découvrir sur des séquences qui n'ont pas la fonction liée au groupe de séquences référence, ces séquences seront appelées faux positifs [8].
- par contre , s'il est trop fin, certaines séquences qui possèdent la fonction liée au motif ne seront pas découvertes , on les qualifiera de vrai négatifs [8].

3.4. Prédiction de structures

Le nombre de structures primaires possibles pour les protéines est exponentiel en fonction de la longueur ℓ [7]. En revanche le nombre de combinaisons de structures tridimensionnelles est beaucoup plus réduit[7]. Ainsi, des séquences très différentes peuvent avoir des structures similaires. La structure tridimensionnelle d'une séquence est une information contenue dans sa structure primaire. Cependant, cette information est actuellement difficile à déterminer sans utiliser une analyse directe de la séquence. Connaitre la structure primaire d'une protéine ne permet pas actuellement d'en déduire sa structure tridimensionnelle [7]**.**

La structure 3D d'une protéine peut être déterminée expérimentalement par cristallographie ou par résonance magnétique nucléaire . Ces méthodes sont toutefois assez lourdes à mettre en œuvre, et nécessitent un matériel spécialisé . La prédiction de structures est une branche de la bioinformatique qui consiste à essayer de déterminer la structure d'une protéine sans passer par la phase expérimentale [7].

La méthode qui donne les meilleurs résultats actuellement procède par homologie, c'est-`a-dire en se basant sur des séquences ayant des structures primaires assez proches, et dont la structure tridimensionnelle est déjà connue . Il s'agit l`a d'une application directe du problème d'alignement de séquences. Il est nécessaire pour cela de trouver des séquences similaires [7].

4. Représentation informatique

Nous abordons dans cette section le point de vue informatique pour la représentation des données biologiques [7]. Les notions vues à la section précédente peuvent être formalisées. Il devient ainsi possible de les représenter aisément pour un traitement informatique [7].

4.1. Séquence

On appelle séquence S sur un alphabet $\sum$ une suite ordonnée d'éléments appartenant à $\sum, S = (x_1, x_2, \dots x_n)$ [7].

4.2. Alphabet

On appelle alphabet tous ensemble fini $\sum$ des symboles distincts deux à deux, ainsi $l'ADN$ et $l'ARN$ sont représentés par un ensemble de quatre lettres ($A - T - C - G$ pour ADN) ($A - U - C - G$ pour ARN) et les protéines avec un ensemble de 20 $lettres$ [7].

4.3. Sous séquence

Soit S une séquence de longueur n On appelle sous séquence de S toute partie de S Composée d'un ensemble de caractères consécutifs de S [7].

Nous noterons $S\,[i \ldots j]$ avec $1 \leq i \leq j \leq n$ la sous séquence $S = (x_i, \ldots x_i)$ [7].

4.4. Longueur

Longueur d'une séquence c'est le nombre d'éléments qui la composent $|S| = n$,[7].

5. Format de séquence

Il existe de nombreux formats des séquences : plus d'une trentaine sont répertories et utilise. Nous citons quelque format :

5.1. Fasta format

Format qui permet de représenter un ou plusieurs séquences (nucléiques ou protéiques) [11]. Une ligne qui commence par le symbole $'>'$ caractérise le début d'une nouvelle séquence. Le symbole $'>'$ est suivi d'un identifiant de séquence et des commentaires éventuels [11]. Les lignes suivantes constituent la séquence (jusqu'à ce qu'une nouvelle ligne commence par $'>'$ ou la fin de fichier) [11].

- Exemple :

$> FBtr0302953\ \ type = mRNA\ ;\ \ loc = 2R\ ;\ \ name = CG42703 -$

RARAHHICITAWTGWCGAASASTSACTWRTYUIHWS

RAHHICITAWTGWCGAASASTSACTWRTYUIHWI

5.2. MSF

Un format entrelacé qui est conçu pour simplifier la comparaison des séquences avec des longueurs similaires [11].

- Exemple :

```
 MSF:   96  Type: P    Check:  7038   ..

Name: 1aab_ oo  Len:   96  Check:  4681  Weight:  10.0
Name: 1j46_A oo  Len:   96  Check:  1914  Weight:  10.0
Name: 1k99_A oo  Len:   96  Check:  8221  Weight:  10.0
Name: 2lef_A oo  Len:   96  Check:  2222  Weight:  10.0

//

1aab_        ...GKGDPKK PRGKMSSYAF FVQTSREEHK KKHPDASVNF SEFSKKCSER
1j46_A       ......MQDR VKRPMNAFIV WSRDQRRKMA LENP..RMRN SEISKQLGYQ
1k99_A      MKKLKKHPDF PKKPLTPYFR FFMEKRAKYA KLHP..EMSN LDLTKILSKK
2lef_A       ........MH IKKPLNAFML YMKEMRANVV AEST..LKES AAINQILGRR


1aab_        WKTMSAKEKG KFEDMAKADK ARYEREMKTY IPPKGE.... ......
1j46_A       WKMLTEAEKW PFFQEAQKLQ AMHREKYPNY KYRPRRKAKM LPK...
1k99_A       YKELPEKKKM KYIQDFQREK QEFERNLARF REDHPDLIQN AKK...
2lef_A       WHALSREEQA KYYELARKER QLHMQLYPGW SARDNYGKKK KRKREK
```

5.3. Clustal

Similaire à MSF, mais comprend une ligne supplémentaire signifiant la qualité de l'alignement entre un ensemble de séquences. Produit et lu par les programmes *clustalx* et *clustalw* [11].

6. Les Banques des données Biologique

Les premières banques de données biologiques sont apparues au début des années 80 sous l'initiative de quelques équipes de recherches. Leur principale mission est de rendre publiques les séquences qui ont été déterminées. Les données biologiques stockées dans ces banques sont des séquences primaires d'ADN, d'ARN et de protéines. Les données peuvent être soumises et consultées par l'intermédiaire du Web. Les séquences stockées dans ces banques sont obtenues de plusieurs manières différentes. Il y a celles isolées à partir d'une cellule, déduites à partir de la séquence nucléique par simple traduction (cas des séquences d'ARN ou protéines) ou encore par génie génétique. Les données stockées doivent être consultées d'une manière significative, et souvent le contenu de plusieurs banques de données doit être consulté simultanément et en corrélation les uns avec les autres. Des langages spéciaux ont été développés pour faciliter cette tâche (tels que le système de récupération de séquence «SRS » et le système « Entrez »). Certaines bases de données fournissent la fonctionnalité d'accès aux séquences mais encore des liens vers d'autres bases de données et les résultats d'analyse déjà obtenus. Par exemple, SWISSPROT contient des séquences de protéine ainsi que des annotations décrivant la fonction d'une protéine. Des structures 3D des protéines sont stockées

dans des bases de données spécifiques. On peut trouver des banques spécialisées pour le stockage des motifs. En outre, des bases de données de la littérature scientifique (telles que PUBMED, MEDLINE) fournissent des fonctionnalités additionnelles, par exemple elles peuvent rechercher les articles scientifiques semblables basés sur l'utilisation de la reconnaissance des mots. Ils ont développé des systèmes d'identification des textes qui extraient automatiquement l'information concernant un sujet tel que la fonction d'une protéine à partir des résumés des articles scientifiques [4].

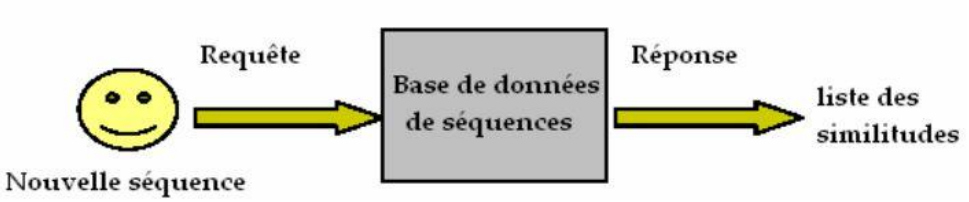

Figure I.3 : *Interrogation d'une base de données*

6.1. Les Banques des Séquences Nucléiques

Nous citons les banques les plus populaires malgré que l'accès soit toujours contrôlé via des mots de passe :

- **EMBL** : banque européenne créée en 1980 et financée par l'EMBO (European Moleculary Biology Organization),elle est aujourd'hui diffusée par l'EBI (European Bio-informatics Institute, Cambridge, UK) [4].
- **GenBank** : créée en 1982 par la société IntelliGenetics et diffusée maintenant par le NCBI (National Center for Biotechnology Information, Los Alamos, US). Elle est soutenue par le NIH (National Institute of Health). Elle possède plus de 50 millions séquences stockées [4].
- **DDBJ (Dna Data Bank)** : créé en 1986 et diffusée par le NIG (National Institute of Genetics, Japon) [4].

La collaboration entre les deux premières banques a commencé relativement tôt. Elle s'est étendue en 1987 avec la participation de la DDBJ. Ils ont adopté un système de conventions communes : "The DDBJ/EMBL/GenBank Feature Table Definition" en 1990 qui a défini un format unique pour la description des caractéristiques biologiques qui accompagnent les séquences dans les banques de données nucléiques [4].

6.2. Les Banques des Séquences Protéiques

PDB : La banque de données sur les protéines du " *Research Collaboratory for Structural Bioinformatics* ", plus communément appelée *Protein Data Bank* ou PDB est une collection mondiale de données sur la structure tridimensionnelle (ou structure 3D) de macromolécules biologiques : protéines, essentiellement, et acides nucléiques. Ces structures sont essentiellement déterminées par cristallographie aux rayons X ou par spectroscopie RMN. Ces données expérimentales sont déposées dans la PDB par des biologistes et des biochimistes du monde entier et appartiennent au domaine public [33]. Leur consultation est gratuite et peut se faire directement depuis les site web de la banque :

- Europe : PDBe ;
- Japon : PDBj ;
- États-Unis : RCSB PDB.

La PDB est la principale source de données de biologie structurale et permet en particulier d'accéder à des structures 3D de protéines d'intérêt pharmaceutique [33].

PIR-NBRF : créée en 1984 par la NBRF (National Biomedical Research Foundation). Elle est maintenant un ensemble de données issues du MIPS (Martinsried Institute for Protein Sequences, Munich, All emagne) et de la banque japonaise JIPID (Japan International Protein Information Database) [4].

SwissProt : créée en 1986 à l'Université de Genève et maintenue depuis 1987 dans le cadre d'une collaboration, entre cette université (via ExPASy, Expert Protein Analysis System) et l'EBI. Celle-ci regroupe aussi des séquences annotées de la banque PIRNBRF ainsi que des séquences codantes, traduites de l'EMBL [4].

6.3. Les Banques de Motif

Prosite : La base de données dédiées aux stockages des motifs protéiques ayant une signification biologique peut être considérée comme un dictionnaire de motifs [4].

Les bases de ce type ont pour mission le recensement dans des catalogues les séquences des différents motifs pour les quels une activité biologique a été identifiée [4].

7. Conclusion

Dans ce chapitre nous avons présenté un introduction sur la bioinformatique. Alors que l'informatique est devenue un apport fondamentale à la biologie moléculaire. Les moyens informatiques sont naturellement utilisés pour le stockage ou la gestion des données mais également pour l'interprétation de ces données . le traitement informatique des séquences peut par exemple déterminer la fonction biologique d'un gène [1]. Cet apport informatique concerne principalement quatre aspects :

- Le premier est l'organisation des données avec essentiellement la création de bases de données afin de réunir le plus d'information possible sur les séquences [1].
- Le deuxième aspect concerne les traitements que l'on peut effectuer sur les séquences afin de repérer un élément biologique intéressant. Ces programmes représentent les traitements couramment utilisés dans l'analyse des séquences comme la recherche des similitudes d'une séquence avec l'ensemble d'une base de données [1].
- Le troisième aspect est celui qui permet d'élaborer des stratégies pour apporter des connaissances biologiques supplémentaires que l'on pourra ensuite intégrer dans des traitements standards [1]. Par exemple la mise au point de nouvelles matrices de substitution des acides amines , etc …
- Enfin, le quatrième aspect est celui de l'évaluation des différentes approches citées précédemment dans le but de valider [1].

CHAPITRE II

Alignement par paires

1. Introduction

L'alignement par paire de séquences de protéines est un outil fondamental de la bioinformatique. Il a pour but principal de faire ressortir les séquences apparentées, en mettant en évidence les régions communes. L'alignement par paire est principalement utilisé pour la comparaison d'une séquence avec un ensemble de séquences. Les algorithmes $Fasta$ et $Blast$ permettent de comparer une séquence à un ensemble de séquences contenues dans une base de données. Une séquence peut être définie comme une suite finie et ordonnée de lettres prises dans un alphabet Σ. Pour les protéines, cet alphabet est constitué de $20\ lettres$, appelées acides aminés ou résidus [10].

2. Définitions

2.1. Définition 1

Soient $S_1 = (x_{11}, x_{12}, \dots, x_{1|S_1|})$ et $S_2 = (x_{21}, x_{22}, \dots, x_{2|S_2|})$ 2 séquences définies sur un alphabet Σ [10]. Un alignement A de S1 et S2 est une matrice de caractères de $\Sigma \cup \{-\}$ définie par [10]:

$$A = \begin{bmatrix} a_{11}, a_{12}, \dots, a_{1q} \\ a_{21}, a_{22}, \dots, a_{2q} \end{bmatrix}$$

et vérifiant les propriétés [10]:

- $\max(|S_1|, |S_2|) \leq q \leq |S_1| + |S_2|$,
- $a_{ui} = x_{uv}\ ou\ -, \forall u \in \{1,2\}, \forall v \in [1..|S_u|]$
- $\nexists i\ tel\ que\ a_{1i} = a_{2i} = -$.

2.2. Définition 2

Soient S_1 et S_2 deux séquences de longueurs respectives m et n définies sur un alphabet Σ. Soient $p(i,j)$ le problème consistant à aligner les deux sous-séquences $S_1[1..i] et\ S_2[1..j]$, $i \leq m\ et\ j \leq n\ et\ D(i,j)$, la distance d'édition associée [10].

- Le problème consiste à déterminer $P(m,n)$.
- Le sous-problème $P(i,j)$ consiste à calculer $D(i,j)$.
- L'initialisation sa fait avec $P(i,0)\ et\ P(0,j)$.

Les problèmes $P(i,0) et\ P(0,j)$ représentent le décalage d'une des séquences par rapport à l'autre, et leurs coûts sont simples à calculer [10].

Exemple d'alignement de deux séquences

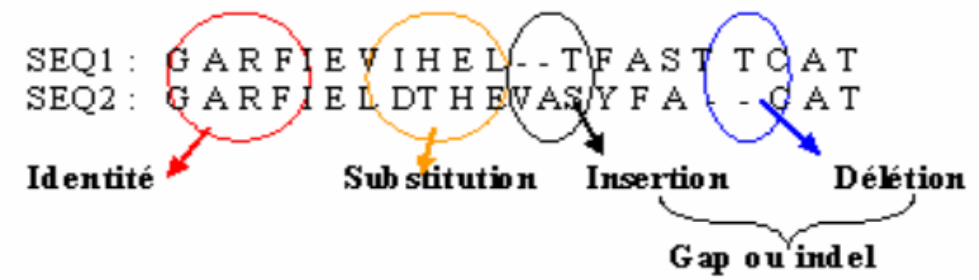

Figure II.1 : *Alignement de deux séquences protéiques [4]*

3. Évaluation d'un Alignement

Cependant, il est clair que pour deux séquences données quelconques il y a plusieurs alignements possibles. Il est devenu alors nécessaire de pouvoir déterminer quel est le meilleur alignement ou plutôt l'optimal si possible. Évaluer un alignement revient alors à mesurer sa qualité en déterminant la distance qui sépare les deux séquences. Le score d'un alignement est la somme des scores de toutes les positions de bases (résidus) prises deux à deux [4].

Exemple d'évaluation :

On peut attribuer une valeur positive à des symboles alignés identiques et une pénalité (valeur négative) à une substitution ou à un gap [4].

Si l'on considère l'exemple précédent :

- Score(identité) =2
- Score(substitution) = -1
- Score(gap) = -2

Le score de cet alignement serait alors :

Figure II.2 : *Score d'un alignement [4]*

Pour évaluer un alignement, le poids de chaque paire de résidus (identité ou substitution) dépend de la nature des résidus mis en correspondance [4]. Le calcul de score d'un alignement de deux séquences A et B de longueur équivalente L est alors :

$$Score\ (A,B) = \sum_{i=1}^{L} SC(A_i, B_i)$$

4. Distance et similarité entre deux séquences

4.1. Similarité et homologie

La similarité entre deux séquences peut être expliquée en prenant comme postulat de départ que toutes les espèces vivantes sont issues d'un même ancêtre. Selon cette théorie, des mutations interviennent au niveau de protéine, générant ainsi de nouvelles espèces. Ces mutations se produisent localement et peuvent être de différents types [7]:

- suppression d'un ou plusieurs acides aminés,
- insertion d'un ou plusieurs acides aminés,
- mutation d'un acide aminé en un autre.

Au sens de la théorie de l'évolution, deux séquences peuvent donc être plus ou moins proches, selon le nombre de modifications ayant eu lieu [7].

- Définition

On dit qu'il y a homologie entre deux séquences lorsque celles-ci possèdent une parenté du point de vue de l'évolution. On dit qu'il y a similarité de séquences, lorsqu'il y a de nombreuses identités entre les séquences. Pour les protéines, cela se caractérise par des paires de résidus composées de deux acides aminés appartenant à la même famille physicochimique[7].

4.2. La distance de Hamming

Soient S et T deux séquences de même longueur n sur un alphabet $\sum$. La distance de $Hamming$ entre S et T, notée $dH(S,T)$, représente le nombre de caractères $S[i]$ et $T[i]$ qui ne se correspondent pas [7]. La distance de $Hamming$ est définie par :

$$dH(S,T) = \left| \left\{ i \in \frac{[1..n]}{S[i]} \neq T[i] \right\} \right|$$

Cette formule compte le nombre de positions ou les lettres ne se correspondent pas [7].

4.3. Le Système de Score

Un système de score est le coût à attribuer aux opérations élémentaires (identité, substitution, délétion et insertion) de comparaisons de séquences. En général, on a besoin donc [4]:

- Des systèmes de scores qui soient « biologiquement pertinent » .
- Des matrices de substitution et donc des scores individuels $SC(A_i, B_j)$, dont le choix dépend de la relation recherchée entre les deux séquences :
  - Relation structurelle (propriétés physico chimiques) .
  - Relation d'homologie (évolution moléculaire).

4.4. Les Matrices de Substitution

Le choix d'une matrice de substitution gouverne le système des scores et par conséquent influe sur les résultats obtenus. Il existe deux types de matrices de substitution à utiliser selon la nature des séquences nucléiques ou protéiques [12].

4.4.1. Matrices de Scores pour l'ADN

- **La matrice Identité :** Cette matrice consiste en l'attribution d'un score 1 en cas d'identité sinon un zéro.

–	A	C	G	T
A	1	0	0	0
C	0	1	0	0
G	0	0	1	0
T	0	0	0	1

Tableaux II.1 : *Matrice de Score Pour l'ADN* [12]

- **La matrice de Transition/Transversion :** Dans cette matrice en prend en considération l'effet des actions des transitions (A à G, G à A,C à T, et T à C) et Transversion (les autres passages entre nucléotides) Identité=3Transition= 1, Transversion = 0.

–	A	C	G	T
A	3	0	1	0
C	0	3	0	1
G	1	0	3	0
T	0	1	0	3

Tableaux II.2 : *Matrice de transition* [12]

- **La matrice BLAST :** La matrice identité Blast. C'est une matrice de même principe que la matrice Identité sauf que les valeurs attrib-uées en cas d'identité et substitution sont différentes de 1 et 0 . On Remarque que la substitution ici est fortement pénalisée[12].

–	A	C	G	T
A	5	-4	-4	-4
C	-4	5	-4	-4
G	-4	-4	5	-4
T	-4	-4	-4	5

Tableaux II.3 :*La matrice BLAST* [12]

4.4.2. Matrices de score pour les protéines

Deux grandes familles de matrices **.**

- **Matrices PAM :**

Les matrices PAM pour ≪ Percent Accepted Mutation / Accepted Point Mutation ≫, sont construites par étude de segments pris dans des séquences protéiques homo-logues (moins de 15% de différences) [13]. PAM x : x % de mut-ations acceptées entre les séquences qui ont servi à construire la matrice. Les fréquences de substitutions observées (ou proba-bilité conditionnelle : appelée "odd") sont transformées en logarithme de probabilité, normalisé en unité d'évolution. Le logarithme est utilisé pour que dans les programmes de recherche de ressemblance, la somme de ces éléments donne le logarithme de la probabilité pour la séquence entière (le modèle étant Markovien : indépendance de fréquences de substitution)[13].

Les éléments diagonaux de la matrice indiquent un Évolution sans substitution. Pour PAM1 , leur somme est telle qu'elle correspond à une probabilité de 99/100 (1 mutation pour 100 résidus : d'où le nom PAM : accepte point mutation) [13].

L'indépendance des fréquences et les éléments de la matrice étant des logarithmes de fréquences, on peut calculer PAM (N) en élevant PAM1 à la puissance N, par exemple : pour PAM120, il faut multiplier PAM1 par elle-même 120 fois [13].

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

Tableaux II.4 : *Matrice PAM 250* [13]

Remarque : la valeur $SC(X_i, Y_j)$ sera :

- S = 0 : les probabilités observées et attendues sont identiques.
- S < 0 : les probabilités observées sont inférieures aux attendues.
- S > 0 : les probabilités observées sont supérieures aux attendues.

- **Matrice BLOSUM :**

Ces matrices BLOSUM (Blocks Substitutions Matrices) sont construites par analyse de séquences de protéines. Les séquences sont découpées en blocs (2000résidus au total) par rapport au pourcentage d'acides aminés inchangés [14]. BLOSUM x : matrice obtenue à partir de séquences présentant au minimum x % d'identité (similitude) entre elles [14]. Une

matrice "odds" est calculée à partir des blocs d'alignement pour chaque valeur de similitude, et ensuite chaque élément est transformé en unité d'information en prenant le logarithme du rapport de la valeur observée à la valeur qu'on obtiendrait au hasard. Cette matrice est ensuite normalisée [14]. Les correspondances entre BLOSUM et PAM, basées sur la théorie de l' information sont :

- $PAM250 -> BLOSUM45$
- $PAM160 -> BLOSUM\ 62$
- $PAM120 -> BLOSUM\ 80$

A	4																			
C	11	5																		
D	-2	0	6																	
E	-2	-2	1	6																
F	0	-3	-3	-3	9															
G	-1	1	0	0	-3	5														
H	-1	0	0	2	-4	2	5													
I	0	-2	0	-1	-3	-2	-2	6												
K	-2	0	1	-1	-3	0	0	-2	8											
L	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
M	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
N	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
P	-1	-1	-2	-3	-1	0	-1	-3	-2	1	2	-1	5							
Q	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
R	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
V	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	-1	-4	-2	-2	11		
W	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Y	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4
	A	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**K**	**L**	**M**	**N**	**P**	**Q**	**R**	**S**	**T**	**V**	**W**	**Y**

Tableaux II.5 : *Matrice BLOSUM 62* [13]

4.5. Evaluation des brèches

L'évaluation des brèches dans un alignement est très importante. Selon les valeurs attribuées, le nombre des brèches peut varier. Si elles ne sont pas assez pénalisantes, la fonction de score risque de favoriser les alignements qui en contiennent beaucoup. En particulier, en morcellent les séquences il est possible d'augmenter le nombre de correspondances [15].

Inversement, si les brèches sont trop fortement évaluées, les alignements ne comportant pas assez des brèches sont favorises, empêchant ainsi d'avoir toutes les correspondances. Il existe principalement deux modélisations pour évaluer le cout engendré par l'insertion d'une brèche. Les valeurs associées à une brèche pour ces modelés peuvent être obtenues par des fonctions. Ces fonctions prennent en paramètre la longueur de la brèche et retourne le cout de celle-ci [15].

4.5.1. Les brèches à cout constant

Ce modèle permet une évaluation très simple de l'alignement. En effet, comme toutes les positions d'une brèche ont la même valeur, l'évaluation repose sur le même principe que pour les paires de résidus. Il suffit pour cela de faire la somme de toutes les valeurs sur la longueur de l'alignement. Le symbole '−' peut alors être considère comme une acide amine pour l'évaluation [7].

4.5.2. Les brèches à cout affine

Selon les biologistes, l'évaluation a cout constant même si elle est simple à mettre en œuvre a un défaut majeur. Elle n'est en effet pas représentative de la façon dont les choses se passent dans la réalité. D'un point de vue biologique, la création de la brèche est beaucoup plus pénalisante que son élongation. Il semsble alors normal de prendre en compte cette remarque importante, et donc d'associer un cout diffèrent suivant qu'il s'agit du début ou de l'extension de la brèche. Les dénominations généralement utilisées pour le cout d'ouverture d'une brèche sont K ou gop (opening penalty). Pour l'extension de la brèche on utilise fréquemment h ou gep (extending penalty). Le cout associe à une brèche de longueur l est alors la fonction affine $gap(l) = K + h \times (l - 1)$. On trouve également cette fonction sous la forme $gap(l) = K' + h \times l$ en posant $K' = K - h$ [7].

L'évaluation d'un alignement au moyen de cette méthode ne peut se faire de la même façon qu'avec les brèches à cout constant . Il est possible de réaliser le calcul suivant deux méthodes :

- La première consiste à n'évaluer que les paires des résidus, puis à y ajouter la somme des valeurs de toutes les brèches [7].
- La seconde méthode consiste à évaluer l'ensemble de l'alignement en faisant la somme de toutes les positions. Cette méthode nécessite lorsque l'on doit évaluer une brèche de savoir si la position précédente était également une brèche ou

non. En pratique, seule cette méthode est utilisée pour la construction de l'alignement, même si elle génère plus de calculs que dans le cas des brèches à cout constant [7].

Les valeurs généralement utilisées pour des matrices de substitution standard sont de l'ordre de -10 pour K et -1 pour h. Ces valeurs peuvent bien sur varier, mais en conservant toujours un ratio K/h voisin de 10 [7].

5. Principe de la programmation dynamique

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

5.1. Problème et sous-problème

Soit $P(n)$ un problème d'optimisation de taille n. On appelle sous-problème de $P(n)$ tout problème $P(i)$ avec $i < n$.

5.2. Principe d'optimalité

On dit qu'un problème $P(n)$ satisfait au principe d'optimalité lorsqu'une solution optimale peut être exprimée en fonction de solutions optimales de sous-problèmes [7].

5.3. Programmation dynamique

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 [7].

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" [7].

6. Les Méthodes d'Alignement par paires

Il existe deux types d'alignements de séquences : global et local.

Le premier prend en considération l'ensemble des résidus de chacune des séquences. Si les longueurs des séquences sont différentes, alors la plus courtes va subir des insertions de gaps afin d'arriver à aligner les deux séquences d'une extrémité à l'autre. Cependant dans un alignement global, si uniquement des segments courts sont très similaires entre deux séquences, les autres parties des séquences risquent de diminuer le poids de ces régions. C'est pourquoi d'autres algorithmes d'alignements, dits locaux, basés sur la localisation des zones de similarité sont nés. Le but de ces alignements locaux est de trouver sans prédétermination de longueur les zones les plus similaires entre deux séquences. L'alignement local comporte donc une partie de chacune des séquences et non la totalité des séquences comme dans la plupart des alignements globaux [4].

6.1. Alignement Global

Plusieurs méthodes ont été développées afin de réaliser un alignement global de deux séquences le plus correct que possible. Parmi ces méthodes et qui sont toujours utilisées on trouve des méthodes graphiques et autres qui utilisent la programmation dynamique [4].

- Algorithme Needleman & Wunsch

Basé sur la programmation dynamique (la récursivité), cet algorithme ne calcule pas la différence entre deux séquences mais la similarité. Considérons deux séquences $A(1,n)$ $B(1,m)$ [4].

Un tableau à deux dimensions est rempli ligne après ligne (en partant de la dernière) et pour chaque ligne , colonne après colonne (en partant de la dernière) en obéissant à la règle suivante [4]:

Le score $S(i,j)$ est le nombre maximum de correspondance entre les deux parties de séquences $A(i,n)$ et $B(j,m)$ (en prenant tous les chemins possibles à partir de $(i\ ,j)$) et en appliquant une fonction de score [4]:

- score pour une identité =1 .
- score pour une substitution, une insertion ou délétion =0 .

La formule de récurrence est [4]:

$$S(i,j) = \max \begin{cases} \text{Si } ai = bj + 1 \quad S(i, j+1) - 1 + s(ai, bj), \quad \text{sinon } S(i, j+1) + s(ai, bj) \\ \text{Si } ai+1 = bj+1 \quad S(i+1,j+1) - 1 + s(ai, bj), \quad \text{sinon } S(i+1,j+1) + s(ai, bj) \\ \text{Si } ai+1 = bj \quad S(i+1,j) - 1 + s(ai, bj), \quad \text{sinon } S(i+1, j) + s(ai, bj) \\ \text{Avec évidemment} \quad S(n+1, j) = S(i, m+1) = 0 \end{cases}$$

La similarité entre les deux séquences est égale à la valeur de $S(1,1)$ et l'alignement est un graphe qui a pour origine $S(1,1)$ et parcourt la matrice pour des i et j croissant en recherchant l'élément maximal voisin [4].

6.2. Alignement Local

Ce type d'alignement est favorable aux séquences divergentes car un alignement global serait non significatif. Pour l'alignement Local, Smith et Waterman une méthode exacte qui permet d'aligner deux séquences en essayant d'aligner des segments communs ou motifs [4].

- L'algorithme Smith &Waterman

L'algorithme de Smith et Waterman est décrit pour l'alignement local de deux séquences. Il identifie les sous séquences maximales de deux séquences par programmation dynamique [4].

La différence essentielle de cet algorithme avec l'algorithme de Needleman et Wunsch est que n'importe quelle case de la matrice de comparaison (initiale) peut être considérée comme point de départ pour le calcul des scores sommes et que tout score somme qui devient inférieur à zéro stoppe la progression du calcul des scores sommes et il sera réinitialisé par la valeur 0. La case concernée peut être considérée comme nouveau point de départ. Cela implique que le système de score choisi possède des scores négatifs pour les mauvaises associations qui peuvent exister entre les éléments des séquences [4].

L'équation utilisée pour le calcul de chaque score somme pendant la transformation de la matrice initiale prend alors l'expression suivante[4]:

$$S(i,j) = \max \begin{cases} Sc(i,j) + S(i+1, j+1) \\ Sc(i,j) + \max S(x, j+1) - P \\ Sc(i,j) + \max S(i+1, y) - P \\ 0 \qquad \text{avec } i+2 < x < m \text{ et } j+2 < y < n \end{cases}$$

Où $S(i,j)$ est le score somme de la case d'indice i et j, Sc le score élémentaire de la case d'indice i et j de la matrice initiale issu d'une matrice de substitution et P la pénalité donnée pour une insertion [4].

7. Comparaison avec les Banques de Séquences

Lorsque l'alignement d'une séquence est réalisé contre une banque, le problème du temps de calcul devient prédominant. Les algorithmes précédents sont trop gourmands en ressource. La façon de voir la ressemblance a été posée d'une manière différente pour contourner cet obstacle et des heuristique sont été proposées [4].

8. Les algorithmes (Fasta et Blast)

8.1. Fasta (Fast alignement)

L'algorithme est basé sur l'identification rapide des mots communs entre la séquence à analyser et les séquences de la banque. La taille du mot recherché est un paramètre que l'on peut choisir entre 1 et 6 (2 pour les protéines, 4 à 6 pour l'ADN). FASTA cherche les mots en se basant sur la technique de Dotplot où il identifie les diagonales ayant le plus grand score et en prenant en considération les pénalités de mésappariement. Les scores dix meilleures diagonales vont être recalculés en utilisant une matrice PAM250. Les dix meilleures [4].

Diagonales vont êtres alors rattachées pour former une seule séquence en tenant compte des insertions et délétions [4].

Ce sont les scores des diagonales qui vont être utilisés pour classer les séquences de la base. Les recherches peuvent être optimisé en appliquant un des algorithmes Needleman-Wunsch ou Smith-Waterman mais seulement sur la zone contenant les dix diagonales [4].

Cet algorithme a toutefois un inconvénient : des régions de faible homologie peuvent être ajoutées [4].

8.2. Blast (Basic Local Alignment Search Tool)

Contrairement à FASTA, l'algorithme BLAST ne sélectionne que les séquences qui contiennent des régions de grande similitude. En plus, l'algorithme est basé dans sa conception sur un modèle statistique. L'unité fondamentale de BLAST est le HSP (Highscoring Segment Pair). C'est un couple de segments identifiés sur chacune des séquences comparées, de longueur égale mais non prédéfinie [4].

Toutefois, cet algorithme présente une limite : pour une longueur de segment égale à n (n fixé) une séquence est similaire à 90% par rapport à la séquence de référence, avec un mésappariement tous les n acides aminés, elle ne sera pas repérée. Les versions postérieures de BLAST viennent pour régler ce problème, en prenant en compte les délétions et en ajoutant des pénalités en cas d'ouverture de gap, et même en cas d'extension de celui ci. BLAST2 est plus trois fois plus rapide que son aîné [4].

9. Conclusion

Les méthodes d'alignement par paires dans ce chapitre sont utilisées pour comparer des séquences deux à deux. Elles sont utilisées pour rechercher une homologie entre une séquence test et une séquence de référence, souvent extraite d'une base de donnée. Elles sont les plus simples à mettre en œuvre, et ce sont les seules pour lesquelles il existe des solutions algorithmiques optimales, basées sur la programmation dynamique. Il existe également des méthodes heuristiques rapides, qui permettent d'effectuer des recherches systématiques dans les banques de séquence. Dans ce cas, on compare une séquence inconnue à toutes les séquences de la base, en les testant successivement une par une. Les méthodes les plus connues sont :

- **l'algorithme de Needleman-Wunsch** qui réalise l'alignement global optimal entre deux séquences,
- **l'algorithme de Smith-Waterman** qui réalise un alignement local optimal.

CHAPITRE III

Alignement multiple des séquences

1. Introduction

Le problème d'alignement multiple correspond à une généralisation de l'alignement par paire avec $k > 2$ séquences. Il ne s'agit plus en revanche ici de détecter une simple similitude entre séquences. L'alignement multiple de séquences est utilisé pour différentes opérations. Il permet de déterminer des sous-groupes de séquences en fonction du degré de similarité. Ce qui constitue le point de départ pour la reconstruction de phylogénie [10].

L'alignement multiple permet également de mettre en évidence les zones conservées dans un ensemble de séquences. En partant du principe que des motifs similaires induisent des fonctions identiques, l'alignement multiple permet de prédire la fonction de protéines inconnues en les alignant avec des protéines connues [10].

2. Définition

Soit $S = \{S_1, \ldots, S_k\}$ un ensemble de séquences définies sur un alphabet Σ. Un alignement multiple de S est une matrice d'éléments de $\Sigma \cup \{-\}$ définie par [10]:

$$A = \begin{bmatrix} a_{11}, a_{12}, \ldots, a_{1q} \\ \ddots \\ a_{k1}, a_{k2}, \ldots, a_{kq} \end{bmatrix}$$

Et vérifiant les propriétés :

- $Max_i(|S_i|) \leq q \leq X_i|S_i|$,
- $a_{ui} = x_{uv} ou -, \forall u \in \{1..k\}, \forall v \in [1..|S_u|]$,
- $\nexists j \; tel \; que \; \forall i, a_{ij} = -$.

3. Les utilisation en bioinformatique

L'alignement multiple de séquences permet de mettre en évidence les similarités entre plusieurs séquences. Il est donc possible de comparer simultanément la proximité de toutes ces séquences. Les informations apportées par ces comparaisons permettent d'obtenir des renseignements importants sur les séquences comme les distances d'une séquence par rapport aux autres ou encore la mise en évidence de zones identiques entre plusieurs ou toutes les séquences [7].

Or ces opérations sont très employées en bioinformatique pour la résolution de plusieurs problèmes. L'alignement multiple de séquence est donc principalement utilisé

comme opération préalable pour ces déférents problèmes. Citons par exemple la construction de phylogénie, le prédiction de structure 3D, la détermination de fonction des protéines [7].

4. Évaluation d'un alignement multiple de séquence

Le critère d'évaluation utilisé est souvent une fonction de score. Soit f une fonction permettant de définir la qualité d'un alignement multiple. f est une fonction définie sur Σ'^2 et à valeurs dans R. Nous donnons dans la section suivante quelques exemples des fonctions les plus utilisées [7].

Le nombre d'alignements multiples possibles pour l'ensemble des séquences de S est très important. Il est donc nécessaire de pouvoir évaluer la qualité de chaque solution. Quelques fonctions ont été définies afin de permettre une évaluation des alignements, mais elles peuvent également être utilisées par des algorithmes lors de la construction de ces alignements[7].

5. Les fonctions d'évaluation

5.1. La somme des paires

Nous avons donné au chapitre précédent une fonction permettant d'évaluer un alignement de deux séquences. De même que nous avons généralisé la définition d'alignement de deux séquences à un alignement de k > 2 séquences, cette fonction peut également être généralisée[7].

- Définition :

Soit S un ensemble de k séquences, et soit A un alignement multiple de S de longueur l. Soit f une fonction permettant d'évaluer un couple de résidus, on définit la fonction de somme des paires par [7]:

$$Som(A) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{p=1}^{l} f(S_i(p), S_j(p))$$

Cette définition de Som dépend de la fonction f permettant d'évaluer une paire de résidus. f est donc définie sur $\Sigma' \times \Sigma'$ et est à valeurs dans R. La fonction dépend donc de la matrice de substitution utilisée, mais elle dépend également de la méthode utilisée pour évaluer les brèches. Même si en pratique les brèches sont très souvent évaluées avec la méthode de cout affine, il est également possible d'utiliser un cout constant [7].

On constate aisément qu'en associant la valeur 0 au couple $(-,-)$, la définition donnée ci-dessus peut être réécrite sous forme d'une somme d'évaluations d'alignements de deux séquences [7]:

$$Som(A) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} Eval(A(i,j))$$

Ou $A(i,j)$ correspond à l'alignement des deux séquences S_i et S_j [7].

L'avantage de la fonction de somme des paires est qu'elle permet d'évaluer les zones correspondant à un bon alignement en leur donnant des valeurs importantes. En effet, la forme même de la fonction permet de sur évaluer les colonnes contenant de nombreux appariements. Pour une colonne contenant deux occurrences de la même lettre, soit v la valeur indiquée dans la matrice de substitution M. Si dans une colonne il y a α occurrences de cette même lettre, la valeur associée est égale à $\alpha.(\alpha - 1).v/2$. La valeur relative d'une colonne contenant peu d'appariements est donc négligeable devant une colonne qui en contient beaucoup [7].

5.2. La somme des paires pondérées

La fonction de somme de paires que nous venons de présenter est simple à mettre en œuvre, et elle permet d'évaluer la qualité d'un alignement multiple. La raison principale de la pertinence de cette fonction est liée au terme en α^2. En effet lorsque les séquences alignées sont très similaires, la fonction donne des valeurs très élevées lorsque l'alignement est de bonne qualité [7].

Pourtant sous cette forme, la fonction peut avoir un défaut important. Lorsque toutes les séquences sont très similaires, la fonction est tout à fait adaptée. En revanche, lorsque ce n'est pas le cas cela peut poser des problèmes . Un exemple fréquent que l'on peut trouver dans les jeux de séquences est celui des séquences dites orphelines. Lorsque sur un ensemble de séquences à aligner, toutes sauf une sont similaires, la séquence différente est dite orpheline. Sa valeur au sein de l'alignement est négligeable par rapport aux autres séquences, la façon dont cette séquence est alignée n'a que très peu d'influence sur le résultat que donne la fonction. On peut remédier à ce problème en multipliant par un coefficient supérieur à un toutes les évaluations associées à cette séquence [7].

L'exemple de la séquence orpheline est très représentatif, mais d'une façon plus générale l'′evaluation par la fonction de somme des paires est biaisée d`es qu'il y a un ou plusieurs

sous-ensembles de séquences très similaires. Il existe donc une variante permettant de corriger en partie ce défaut. Le principe est assez simple : pondérer les séquences pour pouvoir donner plus d'importance à celles qui sont les plus distantes. Ainsi, il devient possible de pondérer chaque alignement par paire de la somme des paires, en donnant d'autant plus de poids que les séquences sont distantes des autres [7].

- Définition :

Soit S un ensemble de k séquences, soit A un alignement multiple de S de longueur l, et soit wi le poids la séquence S_i . On définit la fonction de somme des paires pondérée par [7]:

$$Som(A) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} w_i . w_j . A(i,j)$$

La détermination du poids $w_i$pour chaque séquence dépend de sa distance par rapport aux autres. Les valeurs doivent donc être déterminées pour chaque problème. La méthode généralement utilisée est basée sur l'algorithme du Neighbour-Joining. Le principe est assez simple, et sera exposé plus avant dans le chapitre. Cet algorithme est en effet utilisé lors de la construction d'un guide-tree et nous verrons comment calculer le poids de chaque séquence[7].

5.3. La fonction Coffee

L'algorithme Coffee propose une méthode différente pour évaluer les alignements multiples. La méthode utilise ici aussi les alignements par paires, mais pas de la même façon. La fonction d'évaluation prend en compte le fait que l'algorithme permettant de déterminer le meilleur alignement de deux séquences est un algorithme exact.

- Définition :

Soit $S = \{S_1, S_2, \ldots, S_k\}$, un ensemble de $k > 2$ séquences, et soit $A = \{S'_1, S'_2, \ldots, S'_k\}$ un alignement multiple de S. L'évaluation au moyen de la fonction Coffee peut se décomposer en deux étapes :

- Dans un premier temps, l'algorithme commence par réaliser tous les alignements par paires des séquences de S,
- L'évaluation se fait au moyen des alignements obtenus à la première étape. En effet,

- l'alignement par paire A_{ij} de chaque couple de séquences (S'_i, S'_j) est comparé à l'alignement par paire S_{ij} des séquences S_i et S_j [7].

Cette évaluation se fait en comparant les paires de résidus de A_{ij} et S_{ij}. Ainsi, l'alignement A_{ij} est parcouru, et pour chaque paire de résidus, une valeur est attribuée. Si cette paire de résidus est présente dans l'alignement S_{ij} elle a la valeur 1, sinon elle a la valeur 0. L'évaluation de A_{ij} se fait en additionnant les valeurs de chaque paire de résidus, celle-ci est notée $SCORE(A_{ij})$ [7].

L'évaluation de A au moyen de la fonction Coffee est obtenue en additionnant les valeurs de tous les couples de séquences A_{ij} de A. Pour pallier le problème de la fonction de somme des paires, les valeurs sont ici aussi pondérées. Nous obtenons donc la valeur suivante [7]:

$$f(A) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{j=k} w_i . w_j . SCORE(A_{ij})$$

Cette fonction permet de comparer les valeurs associées à plusieurs alignements possibles de S, mais elle ne permet pas de connaitre la qualité réelle de ces alignements. Ainsi la longueur des séquences a autant d'influence sur la valeur de cette fonction que la qualité de l'alignement. Pour éviter cela Coffee est définie à partir de la fonction donnée ci-dessus, en la divisant par la somme des longueurs des alignements par paires de A [7]:

$$Coffee(A) = \frac{\sum_{i=1}^{k-1} \sum_{j=i+1}^{j=k} w_i . w_j . SCORE(A_{ij})}{\sum_{i=1}^{k-1} \sum_{j=i+1}^{j=k} w_i . w_j . Long(A_{ij})}$$

Ou Long est la fonction qui a un alignement de séquences associe sa longueur [7].

Si toutes les paires de résidus de l'alignement multiple A sont identiques aux paires de résidus des alignements par paires correspondants, cela veut dire que pour tout i et j $SCORE(A_{ij}) = \text{Long}(A_{ij})$. Par conséquent, la fonction peut se simplifier, et on obtient $Coffee(A) = 1$ [7].

A l'opposé, si aucune paire de résidus de A_{ij} n'est présente dans l'alignement S_{ij} quels que soient i et j , alors le numérateur de la fonction Coffee est toujours nul, et donc $Coffee(A) = 0$. Coffee fournit donc comme évaluation un pourcentage de similitude entre les alignements par paires des séquences de S et les paires de séquences alignées de A .

Comme l'hypothèse de d´épart était que les alignements par paires des séquences de S sont tous exacts, plus la valeur de $Coffee(A)$ est proche de 1, meilleur est l'alignement A [7].

Le principal avantage de la fonction Coffee est qu'elle offre une méthode d'évaluation qui est plus proche du problème lui-même. La matrice de substitution et la fonction d'´evaluation ne sont utilisées que pour réaliser les alignements par paires [7].

6. Les approches d'Alignements Multiple de Séquences

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 [5].

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 [5].

Les méthodes qui suivent l'approche progressive, sont reconnues d'être très rapides et donnent des résultats assez satisfaisants mais leur inconvénient est le fait de s'arrêter sur les minima locaux et si une erreur est commise au début de l'alignement, elle va se propager sur l'alignement final [9].

L'approche itérative est une manière très simple, rapide et efficace permettant d'améliorer des méthodes d'alignement multiples. L'itération peut être employée pour améliorer le résultat d'un logiciel existant avec n'importe quelle fonction objectif. Elle peut également être incorporée à une stratégie progressive d'alignement pour établir des alignements à partir de zéro pour produire encore de meilleurs résultats [17].

6.1. L'Approche Exacte

L'approche exacte n'est autre qu'une généralisation des méthodes de programmation dynamique de La méthode de programmation dynamique utilisée pour aligner deux séquences, a été appliquée à l'alignement de plusieurs séquences (N dimensions) tels que MSA [6] et DCA [41].

Ce type de méthodes représente de gros problèmes : Le temps de calcul et l'espace mémoire [19].

- 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

- Exemple :

pour 10 séquences de 100 résidus , et 10^{-9} secondes de temps de calcul par colonne , nécessite alors : Temps total = $2^{10} \times 100^{10} \times 10^{-9} \approx 10^{14} s (> 9^{10} années)$ Espace mémoire : 10 11 6 GB [19].

Le problème de l'alignement multiple exacte a été démontré être un problème NP-complet . D'où le recours aux méthodes approchées ou heuristiques [19].

6.2. L'Approche 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 [19]..

6.3. 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 [20]:

Un alignement multiple progressif suit les étapes suivantes [20]:

- Alignement deux à deux de toutes les séquences.
- Construction d'une matrice de distances entre toutes les séquences.
- Détermination de l'ordre selon lequel les séquences seront alignées en utilisant la notion de clustering :
  - Alignement de deux séquences .
  - Alignement d'une séquence et d'un profil .

- Alignement de deux profils .

Problèmes majeurs des alignements multiples progressifs [20]:

- Les alignements entre sous-groupes sont gelés. Si une erreur est produite au début, aucune modification ou correction ultérieure n'est possible .
- Les erreurs dans les alignements des sous-groupes initiaux se propagent dans tous l'alignement.

7. Classification des Méthodes

Les algorithmes d'alignement multiples de séquences sont très nombreux, ils peuvent prendre des formes très différentes et être basés sur des principes très différents .Il est toutefois possible de les regrouper selon trois classes [36] en fonction de critères assez simples. Il existe en effet quelques algorithmes exacts permettant de réaliser des alignements d'un petit nombre de séquences. Les algorithmes approchés sont bien sur beaucoup plus nombreux, et ils peuvent être également subdivisés en deux catégories : les algorithmes progressifs et les algorithmes itératifs . Les algorithmes progressifs ont tous un point commun lié au processus d'alignement. Les séquences sont alignées progressivement en suivant un ordre défini, seule la façon dont vont être alignés les groupes de séquences va différer. Pour les algorithmes itératifs toutes les séquences sont alignées simultanément et les méthodes sont en revanche très différentes les unes des autres. Il est possible de décomposer à nouveau, suivant qu'il s'agit ou non d'algorithmes stochastiques [7].

7.1. L'approche exacts

Le problème d'alignement multiple de séquences est possible de généraliser la méthode basée sur la programmation dynamique. Sa forte complexité spatiale la rend inutilisable en pratique pour la plupart des problèmes d'alignement .Comme nous le montrerons plus loin, à partir de 4 séquences, la quantité de mémoire nécessaire pour réaliser les calculs est bien souvent supérieure à ce qui est disponible dans la plupart des ordinateurs actuels. Aligner 5 séquences est considéré comme étant un petit problème, il est pourtant totalement Exclu de chercher à le résoudre de façon exacte par la méthode la programmation Dynamique que nous avons exposée . Nous présentons donc également ici deux algorithmes Basés sur la programmation dynamique et qui utilisent chacun une heuristique. Le premier Peut aligner une dizaine de séquences, et le second une vingtaine [7].

7.1.1. Méthode basé sur la programmation dynamique

Dans le chapitre précédent nous avons vu comment aligner deux séquences en utilisant la programmation dynamique [7]. Nous allons montrer maintenant que cet algorithme peut être étendu à plus de deux séquences.

Cas de 3 séquences :

Le cas de 3 séquences est le plus simple pour expliquer comment réaliser un alignement multiple en utilisant le principe de la programmation dynamique. L'explication donnée ici et que nous allons principalement détailler concerne le cas le plus simple, à savoir l'alignement avec brèches à cout constant. Le problème consiste ici à aligner 3 séquences S, T et U de longueurs respectives p, q et r. En faisant le parallèle avec le cas de 2 séquences, nous appellerons $P(i,j,k)$ le problème consistant à aligner les 3 sous séquences $S[1\ldots i], T[1\ldots j]$ et $U[1\ldots k]$, et la valeur associée à ce problème sera notée $V(i,j,k)$. Aligner S, T et U consiste donc à déterminer $P(p,q,r)$ et la valeur de cet alignement est $V(p,q,r)$. Le principe de l'algorithme est le même que pour due séquences il suffit de chercher à déterminer la dernière position de l'alignement. Plusieurs cas sont alors possibles, selon qu'il y a une, deux ou trois lettres [7].

- La dernière position de l'alignement constituée d'une unique lettre peut se produire de trois façons différentes, selon la séquence où se trouve la lettre [7].
- La dernière position de l'alignement constituée de deux lettres peut se produire de trois façons différentes, selon la séquence où se trouve la brèche[7].
- La dernière position de l'alignement constituée de trois lettres correspond à un cas Unique. Il y a donc au total 7 possibilités pour la dernière position de l'alignement. Ces 7 possibilités correspondent aux 7 sous-problèmes directs de $P(p,q,r)$. La valeur de $V(p,q,r)$ Comme pour le cas de l'alignement de deux séquences, le principe de la programmation dynamique peut être utilisé pour résoudre le problème. Toutefois dans ce cas il n'est plus possible d'utiliser une matrice en deux dimensions puisqu'il faut conserver toutes les valeurs $V(i,j,k)$. Il est donc nécessaire d'utiliser un cube de taille $(p+1)\times(q+1)\times(r+1)$. Les cas de base correspondent aux problèmes où une au moins des valeurs i,j ou k est égale à 0 [7]. Le calcul des cas de base peut se ramener aux trois cas dégénérés suivants :

- $i = 0$, ce qui correspond à réaliser l'alignement des deux séquences T et U,
- $j = 0$, ce qui correspond à réaliser l'alignement des deux séquences S et U,
- $k = 0$, ce qui correspond à réaliser l'alignement des deux séquences S et T.

Ces trois alignements sont réalisés sur chacune des trois faces du cube se rejoignant au point d'origine $(0,0,0)$. à partir des cas de base, il devient possible de remplir progressivement toute la matrice. L'algorithme se termine lorsque $V(p,q,r)$ a été calculé. Une fois cette valeur obtenue, il est possible de construire l'alignement correspondant en utilisant la même méthode qu'pour deux séquences. Il suffit pour cela de parcourir la matrice jusqu'à l'origine afin de déterminer quelle suite d'opérations d'édition a été utilisée. Le résultat que l'on obtient est similaire à ce que l'on peut obtenir pour deux séquences, mais en 3 dimensions. La montre un exemple de parcours de la matrice pour construire l'alignement[7].

	L=100	L=200	L=300	L=400	L=500
N=4	100 Mo	1.6 Go	8.1 Go	25.7 Go	62.5 Go
N=5	10 Go	320 Go	2.4 To	10.2 To	31.2 To
N=6	1 To	64 To	729 To	4.1 Po	15.6 Po
N=7	100 To	12.8 Po	218.7 Po	1.6 Eo	7.8 Eo
N=8	10 Po	2.5 Eo	65.6 Eo	655 Eo	3.9 Zo

Tableaux III.1 : *Mémoire nécessaire pour un alignement multiple [7]*

7.1.2. La méthode MSA

Nous venons de voir que la programmation dynamique pouvait théoriquement être utilisée pour aligner un nombre quelconque de séquences. Toutefois en pratique l'algorithme ne peut pas être utilisé en raison de la mémoire qu'il requiert. Cas de 2 séquences L'algorithme MSA propose une heuristique basée sur l'algorithme complet de Needleman-Wunsch. Le principe est simple, et il est facile à comprendre sur un alignement de deux séquences. À savoir que le chemin permettant la construction de l'alignement des deux séquences se situe à proximité de la diagonale. En fonction de la similarité entre les séquences à aligner, il est possible de déterminer avant d'utiliser l'algorithme de programmation dans quelle partie se trouvera l'alignement. Plus la similarité est forte, et plus la zone centrale peut être réduite. Les zones hachurées marquées 1 et 2 correspondent aux valeurs qui n'ont pas besoin d'être calculées.

Même si ces valeurs ne sont pas connues, l'algorithme basé sur la programmation dynamique peut être facilement modifié pour le calcul des valeurs situées à la limite de ces zones [21]. Deux cas sont possibles suivant la zone considérée :

- La limite supérieure de calculs est marquée par la zone 2, et la formule (2.1) n'est plus utilisable [21]. Par exemple pour la coordonnée (i, j), V (i − 1, j) n'est pas défini. Or par hypothèse, nous savons que l'alignement ne peut pas être dans la zone 2, il est donc impossible que V(i,j) soit obtenue à partir de cette valeur. Pour la zone limitrophe supérieure, la formule devient donc :

$$V\ (i,j)\ =\ \max(V\ (i,j\ -\ 1)\ +\ Val\ ('-','t'),\\ V\ (i\ -\ 1,j\ -\ 1)\ +\ Val\ ('s','t'))$$

- De même pour la zone limitrophe inférieure, la formule devient :

$$V\ (i,j)\ =\ max\ (V\ (i\ -\ 1,j)\ +\ Val\ ('s','-'),\\ V\ (i\ -\ 1,j\ -\ 1)\ +\ val\ ('s','t'))$$

Les formules données ici correspondent à un alignement avec coût constant pour les brèches [7]. La formule utilisée pour les alignements avec coût affine peut bien sûr être simplifiée de la même façon.

Cas de $n\ >\ 2$ séquences :

La méthode exposée pour 2 séquences est généralisable pour un nombre quelconque n de séquences. La zone de calculs est toujours une partie de la matrice centrée sur la diagonale issue de l'origine. La représentation graphique est toutefois difficile à réaliser, même pour $n\ =\ 3$.Cette méthode basée sur une méthode exacte est une heuristique. Même s'il est fortement probable qu'elle donne une solution optimale, cela n'est aucunement garanti. Elle offre l'avantage de permettre d'augmenter la limite du nombre de séquences pouvant être alignées. Il est ainsi possible d'aligner jusqu'à une dizaine de séquences similaires [7].

7.1.3. La méthode DCA

- Introduction :

L'algorithme MSA est basé sur une restriction de la zone de calculs autour de la diagonale de la matrice. Il est à la base d'un autre algorithme de type "divide-and-conquer" appelé DCA (Divide and Conquer mulitple sequence Alignment). L'intérêt d'un tel algorithme peut se comprendre facilement. Pour cela nous allons prendre un exemple simple, aligner 4

séquences de longueur 300. D'après le tableau, la mémoire nécessaire pour réaliser l'alignement est de 8,1 Go. Le problème revient à aligner 3 groupes de 4 séquences de longueur 100. Ces opérations n'ayant pas à être faites simultanément, l'espace mémoire nécessaire est donc "seulement" de 100 Mo. En coupant les séquences en sous-séquences de longueur 50, l'espace mémoire nécessaire n'est plus que de 6,3 Mo [22].

- Principe de l'algorithme :

Comme nous venons de le voir il est possible de gagner un facteur 1.000 sur la quantité de mémoire en divisant les séquences en 6 sous-séquences . Et il est bien entendu possible de diviser encore la longueur de chaque séquence pour obtenir autant de problèmes de plus petite taille. Toutefois l'algorithme DCA ne réalise pas uniquement un découpage des séquences, et ce pour une raison très simple. Un découpage aléatoire des séquences pose des problèmes d'alignement puisque l'on impose le début et la fin pour chacun d'eux. Le découpage ne doit donc pas être fait de cette façon. En se basant sur les alignements par paires, l'algorithme DCA détermine les points de découpe de chaque séquence pour former des groupes de sous-séquences à aligner . Une fois tous les alignements de ces groupes de séquences effectués, l'algorithme les assemble pour former le résultat. En réalisant l'alignement des groupes de sous-séquences avec l'algorithme MSA, il est possible de réaliser des alignements de plus de 20 séquences . Nous constatons donc qu'avec deux heuristiques, il est possible d'utiliser la programmation dynamique pour obtenir des alignements comportant nettement plus de séquences. Rien ne garantit toutefois que l'on puisse calculer l'optimum [7].

7.2. L'approche progressif

L'Approche progressif [23] 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 MULTALIGN[24], ClustalW [25], et T-Coffee [26] …etc.

Un alignement multiple progressif suit les étapes suivantes [20]:

- Alignement deux à deux de toutes les séquences.
- Construction d'une matrice de distances entre toutes les séquences.

Détermination de l'ordre selon lequel les séquences seront alignées en utilisant la notion de clustering [20]:

- Alignement de deux séquences.
- Alignement d'une séquence et d'un profil.
- Alignement de deux profils.

Problèmes majeurs des alignements multiples progressifs [20]:

- Les alignements entre sous-groupes sont gelés. Si une erreur est produite au début, aucune modification ou correction ultérieure n'est possible.
- Les erreurs dans les alignements des sous-groupes initiaux se propagent dans tous l'alignement.

Les algorithmes d'alignement multiple progressifs sont basés sur les informations obtenues d'un arbre guide construit au préalable. Cet arbre définit un certain rapprochement entre les séquences (homologie ou similitude). Puis on construit progressivement l'alignement multiple en respectant l'ordre défini par l'arbre. Vu le nombre de méthodes développées, cette approche paraît la plus utilisée malgré ses inconvénients [20].

7.2.1. Clustal

ClustalW est basé sur le principe de l'algorithme de Feng et Doolittle. Le principe général est le suivant [27]:

1. Calculer tous les alignements par paires des séquences, et en déduire une matrice des distances.
2. Construire un guide-tree à partir de la matrice des distances en utilisant la méthode du NeighbourJoining.
3. Utiliser le guide-tree afin de déterminer l'ordre dans lequel les séquences doivent être, Alignées :
    a) Choisir les séquences ou profils à aligner en suivant le guide-tree.
    b) Les aligner en utilisant une méthode basée sur la programmation dynamique.
    c) Créer un profil à partir du résultat de l'alignement.
    d) Si on n'est pas arrivé à la racine de l'arbre reprendre en 3.a .
4. Retourner l'alignement obtenu.

L'alignement de deux séquences et/ou profils dans ClustalW fait intervenir des paramètres supplémentaires par rapport à la méthode exposée au *chapitre II* [27]:

- Ainsi, pour éviter le morcellement des séquences que peuvent créer plusieurs brèches trop rapprochées un paramètre est ajouté. Il s'agit d'une distance minimale devant séparer deux brèches, par défaut cette valeur est égale à 5. IE l reste bien entendu possible d'insérer deux brèches plus proches car cela peut être nécessaire, mais dans ce cas une pénalité est ajoutée au coût de la deuxième brèche.
- ClustalW prend également en compte les propriétés physico chimiques propres aux différents acides aminés. Ainsi si deux acides aminés sont normalement plus difficiles à séparer, un sur coût sera attribué pour l'insertion d'une brèche.
- ClustalW détermine les paramètres d'alignement à utiliser (coût d'ouverture et d'extension de brèches ainsi que matrice de substitution) en fonction des séquences à aligner. ClustalW permet d'aligner plus d'une centaine de séquences, et il donne de bons résultats dans la plupart des cas. De par sa construction, le guide-tree associé à n séquences nécessite de réaliser $n - 1$ alignements pour obtenir l'alignement de toutes les séquences. Le temps de calcul de ClustalW est donc pratiquement linéaire en fonction du nombre de séquences à aligner. Pendant près de vingt ans, ClustalW a été l'algorithme de référence pour l'alignement multiple, et il reste encore aujourd'hui très utilisé. L'algorithme général que nous avons donné peut-être modifié pour donner d'autres algorithmes. Ainsi la fonction de T-Coffee dont nous avons parlé peut-elle être utilisée à la place de la somme des paires pondérée [7].

7.2.2. MUSCLE

La méthode $MUSCLE$ emploie deux mesures de distance pour une paire des séquences: une distance de k-mer de (pour une paire non alignée) et le Kimura distance (pour une paire alignée). Un k-mer est une subséquence 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[16].

La méthode $MUSCLE$ peut être décrite en trois étapes essentielles [16]:

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.
2. 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. 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 ma matrice $D2$. $D2$ va subir le même procédé de clustérisation afin de produire un arbre binaire $TRRE2$ et progressivement construire l'alignement $MSA2$.
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 SP 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 [16].

7.3. L'approche itérative

7.3.1. SAGA

SAGA (Sequence Alignment by Genetic Algorithm) [28] est base sur un algorithme de type génétique . Une population G d'alignements est initialement générée, et le programme permet de contrôler son évolution. La population initiale est crée en générant cent individus par insertion aléatoire de brèches. Le schéma général suivi par l'algorithme est le suivant [29]:

- sélection de la partie de la population devant être remplacée ; par défaut cette valeur est de 50% .
- utilisation de l'un des nombreux operateurs les individus sélectionnés,
- mise à jour de la population en ne conservant que les cent meilleurs individus. Ces trois opérations permettent de passer de la génération G_n à la génération G_{n+1} . L'algorithme s'arrête lorsque la population se stabilise .

Il existe 22 operateurs dans $SAGA$ correspondant à des combinaisons ou à des mutations. Chaque operateur est vu comme une fonction indépendante, prenant deux individus pour les combinaisons et un individu pour les mutations. Tous les opérateurs fournissent en sortie un unique individu [7].

7.3.2. Autre méthode

Il existe plusieurs méthodes et nous parlerons de l'algorithme " Tabu search " .

- Recherche tabou :

L'algorithme recherche tabou procède en deux étapes pour réaliser l'alignement multiple :

1. création d'un alignement initial A_0 par insertion de brèches dans les séquences.
2. utilisation d'une méthode tabou pour déplacer les brèches dans l'alignement. L'ensemble des alignements que l'on peut obtenir en déplaçant une brèche dans A_i constitué les voisins de A_i. L'alignement A_{i+1} est le meilleur voisin qui n'appartient pas à la liste tabou. La solution est obtenue lorsque l'algorithme se stabilise [7].

Plus de détails sur l'algorithme de tabou de recherche dans le chapitre suivant.

8. Les benchmarks

En face de ce grand nombre de méthodes de MSA, il est devenu très difficile voire impossible de dire quelle méthode d'alignement est meilleure qu'une autre [4].

Une évaluation et une comparaison complète des programmes d'alignement exigent un grand nombre d'alignements corrects de référence qui peuvent être employés comme cas de tests [4].

[37] a montré que l'exécution des programmes d'alignement dépend du nombre de séquences, le degré de similitude entre les séquences et le nombre d'insertions dans un alignement. D'autres facteurs peuvent également affecter la qualité d'alignement telle que la longueur des séquences, l'existence de grandes insertions et de prolongements de N/C-terminal. Pour évaluer la qualité d'un alignement, une méthode est répandue actuellement, est l'utilisation des bases de références [4].

Ces bases utilisent un grand nombre d'alignements de référence dits « vrais » comme test. Cette méthode consiste à déterminer la capacité d'un programme d'alignement face à des familles de séquences dont l'alignement optimal est

connu[4]. Plusieurs bases d'alignements de référence ont été élaborées comme BaliBase[38], et OxBench Prefab [39], SABmark [40].

9. Conclusion

dans ce chapitre nous avons présenté le problème de l'alignement multiple des séquences et les fonction d'évaluations, en suite on a abordé les déférant approches de résolution de problème de l'alignement multiple des séquences ainsi que les méthodes de chaque approche enfin on a cité les déférant référence utilisé sur le BALIBASE comme un référence d'évaluation.

CHAPITRE IV

L'algorithme recherche tabou pour l'alignement multiple de séquence

1. Introduction

Dans le chapitre précédent, nous avons parlé du problème de l'alignement multiple des séquences et nous avons vu que le problème est NP-Complet.

Il existe plusieurs algorithmes pour résoudre le problème et sont classés en trois catégories :

- **Les algorithmes exacts** : ils ne peuvent être utilisés que pour un nombre très limité de séquences à cause de la quantité de mémoire nécessaires.
- **Les algorithmes itératifs** : ils donnent en général des résultats acceptables dans un temps de calculs très importants.
- **Les algorithmes progressifs** : ils donnent en général des résultats acceptables dans un temps limité.

Nous avons implémenté dans ce mémoire une approche itérative est appelé l'algorithme recherche tabou pour résolution de problème alignement multiple des séquences, La Recherche Tabou fonctionne à partir d'une solution initial, et explore itérativement le voisinage (neighborhood) de la solution actuel (Current) par générant des mouvements est appelé changer les positions des $gaps$ $('-')$ dans les séquences de chaque solution.

2. La recherche tabou

La recherche tabou (RT) est une métaheuristique à base d'une solution unique. Elle a été proposée en 1986 par Glover [34]. RT est une méthode de recherche locale avancée, elle fait appel à un ensemble de règles et de mécanismes généraux pour guider la recherche de manière intelligente [35]. L'optimisation de la solution avec la recherche tabou se base sur deux astuces: l'utilisation de la notion du voisinage et l'utilisation d'une mémoire permettant le guidage intelligent du processus de la recherche. En parcourant le voisinage de la solution courante S, la recherche tabou ne s'arrête pas au premier optimum local rencontré. Elle examine un échantillonnage de solution du voisinage de S et retient toujours la meilleure solution voisine S', même si celle-ci est de piètre qualité que la solution courante S, afin d'échapper de la vallée de l'optimum local et donner au processus de la recherche d'autres possibilités d'exploration de l'espace de recherche afin de rencontrer l'optimum global. En fait,

les solutions de mauvaise qualité peuvent avoir de bons voisinages et donc guider la recherche vers de meilleures solutions. Cependant, cette stratégie peut créer un phénomène de cyclage (i.e. on peut revisiter des solutions déjà parcourues plusieurs fois). Afin de pallier à ce problème, la recherche tabou propose l'utilisation d'une mémoire permettant le stockage des dernières solutions rencontrées pour ne pas les visiter dans les prochaines itérations et tomber dans le problème du cyclage répétitif. Cette mémoire est appelée « la liste tabou », d'où le nom de la métaheuristique tabou. La taille de la liste tabou est limitée, ce qui empêche l'enregistrement de toutes les solutions rencontrées. C'est la raison pour laquelle la liste tabou procède comme une pile $FIFO$, où la plus ancienne solution sera écartée pour laisser place à la dernière solution rencontrée. Le but de faciliter la gestion de la liste tabou en termes de temps de calcul ou d'espace mémoire nécessaires à pousser les chercheurs à proposer de ne conserver dans la liste que des attributs (i.e. caractéristiques) des solutions au lieu des solutions complètes. Particulièrement, dans le cas où le nombre de variables est très élevé. Toutefois, l'utilisation de la liste tabou peut mener à un blocage dans certains cas. En fait, les nouvelles solutions qui possèdent des attributs des solutions tabou, seront considérées tabou aussi même si elles ne sont pas encore été visitées. Pour remédier à ce problème, on a défini une technique appelée « critère d'aspiration» permettant de sortir du blocage causé par la ressemblance des attributs des solutions tabou. Le critère d'aspiration permet d'omettre le statut tabou sur une solution lorsque certaines circonstances sont respectées. Un critère d'aspiration très classique consiste à autoriser une solution tabou si celle-ci est l'une des meilleures solutions rencontrées depuis le démarrage de la recherche [32].

L'algorithme recherche tabou (schéma général)

Début

Construire une solution initiale s;

Calculer la fitness f(s) de s;

Initialiser une liste tabou vide;

$S_{best} = S'$;

Tant que le critére d'arret n'est pas vérifié **faire**

Trouver la meilleure solution S′ dans le voisinage de S qui ne soit pas tabou ou qui vérifie le critère d'aspiration ;

Calculer f(S′) ;

Si fitness de (S′) est meilleure que fitness de (S_{best}) **alors**

$S_{best} = S'$;

Fin Si

Mettre à jour la liste tabou ;

$S = S'$;

Fin Tant que

Retourner S_{best} ;

Fin

La recherche tabou a été largement utilisée pour la résolution de problèmes d'optimisation difficiles comme: le problème du voyageur de commerce, les problèmes du sac à dos, les problèmes de routage, d'ordonnancement, de coloration de graphes, d'exploitation géologique, etc. C'est une méthode facile à mettre en œuvre, rapide, donne souvent de bons résultats et permet de se sauve r du premier optimum local rencontré contrairement à la méthode de la recherche locale simple. En revanche, la liste tabou demande de ressources importantes si elle est de grande taille. En fait, elle demande une gestion soigneuse de sa taille et de ce qu'elle doit contenir comme informations sur les solutions trouvées, de manière à ne pas être très exigeante en temps de parcours et d'espace mémoire requis et aussi pour éviter les

confusions causées par la ressemblance des attributs des solutions qui bloquent la recherche Il est à noter qu'il existe plusieurs variantes de la recherche tabou. Ces variantes dépendent essentiellement du choix du voisinage et de la manière de gérer la liste tabou [32].

3. Réalisation et expérimentation

Dans ce chapitre nous allons présenter l'environnement de développement et les différents outils utilisés pour réaliser notre algorithme, et expliquer son fonctionnement en présentant notre algorithme de l'alignement multiple de séquences par l'algorithme recherche Tabou, Après cela, nous allons présenter les différents tests et évaluations effectués ainsi que les résultats obtenus par l'algorithme recherche tabou pour prouver leurs importances d'un point de vue de la recherche d'une solution approchée au problème sujet d'étude.

3.1. Environnement de développement

Pour la réalisation de ce travail, nous avons eu recours aux environnements suivants :

3.1.1. Environnement matériel

Pour développer l'application, nous avons utilisé comme environnement matériel :

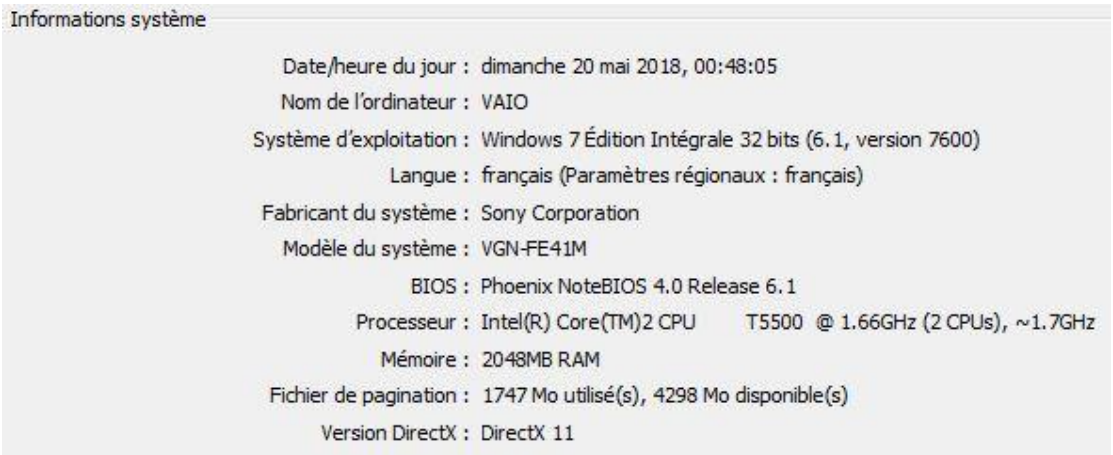

Figure IV.1 : *Environnement matérial*

3.1.2. Environnement logiciel :

- Windows 7 comme système d'exploitation.
- Le NetBeans IDE 8.2 comme Environnement de développement.
- Langage Java.

- NetBeans :

NetBeans Est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web) [30].

Compilé en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Développement Kit JDK est requis pour les développements en Java. NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothéque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme [30].

- Le langage de programmation Java :

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (confondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java. La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que Unix, Windows, Mac OS ou GNU/Linus, avec peu ou pas de modifications, Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java [31].

Le langage Java reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a été épuré des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que les pointeurs et références, ou l'héritage multiple contourné par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc..) [31].

Java permet de développer des application client-serveur. Coté client, les applets sont à l'origine de la notoriété du langage. C'est surtout coté serveur que s'est imposé dans le milieu de l'entreprise grâce aux servlets, le pendant serveur des applets, et plus récemment les JSP (JavaServer Pages) qui peuvent se subtituer à PHP, ASP et ASP.NET [31].

Java a donné naissance à un système d'exploitation (JavaOS), à des environnements de développement (eclipse/JDK), des machines virtuelles (MSJVM(en),JRE) applicatives multiplate-forme (JVM), une déclinaison pour les périphériques mobiles/embarqués (J2ME), une bibliothèque de conception d'interface graphique (AWT/Swing), des applications lourdes (Jude, Oracle SQL Worksheet, etc..), des technologies web (servlets, applets) et une déclinaison pour l'entreprise (J2EE). La portabilité du bytecode Java est assurée par la machine virtuelle Java, et éventuellement par des bibliothèques standard incluses dans un JRE. Cette machine virtuelle peut interpréter le bytecode ou le compiler à la volée en langage machine. La portabilité est dépendante de la qualité de portage des JVM sur chaque OS [31].

3.2. Présentation de l'algorithme

Dans ce problème alignement multiple des séquences, Nous avons appliqué l'algorithme de Recherche Tabou pour aligner les séquences d'acides aminés.

L'aide de la fonction de score (somme des paires), et les matrices de substitution (BLOSUM62 et PAM250), cette algorithme peut être calculer le score de l'alignement à le coût de chaque paire de résidus, et prend une série des séquences non aligner et renvoie une série des séquences aligner représentant la meilleure solution trouvée. On peut le modifier tous les paramètres par défaut de l'algorithme.

3.2.1. Génération de la solution initiale :

La première étape dans un algorithme recherche tabou est de générer la solution (current) initiale . Cela ne se fait qu'une fois chaque fois que le programme est exécuté.

Dans le cas de l'alignement multiple de séquences par l'algorithme recherche tabou, La solution initiale est un alignement différent de la séquence d'entrée.

Pour chaque séquence dans ce alignement. les brèches sont insérées de façon aléatoire pour déterminer toutes les séquences de l'alignement à la même longueur. Cette longueur est à 15% (maximum) de la longueur de la séquence la plus longue dans l'alignement.

3.2.2. La fonction d'évaluation

Les fonction d'évaluation que nous utilisons pour déterminer la qualité d'un alignement est la fonction de somme des paires (SP). Pour déterminer le coût de chaque paire de résidus, nous utilisons la matrice BlOSUM62 ou la matrice PAM250. Le calcul du coût des brèches est effectué suivant le modèle de coût affine, avec comme paramètre par défaut le cout d'ouverture d'une brèche est $gop = -10$, et le cout de l'extension de la brèche $gep = -1$. Ces paramètres peuvent être changés selon le type des séquences en entrées.

$$SP(A_i) = \sum_{i=0}^{k-1} \sum_{j=1}^{j=i-1} SC(s_i, s_j) + Pinalité(gap)$$

Où : $SC(A_i, A_j) = BLOSUM62(A_i, A_j)\ ou\ PAM250(A_i, A_j)$

3.2.3. Structure de voisinage (Neighborhood)

Fondamentalement, la perturbation de la solution actuelle (current) est appliquée pour générer un autre nouvelles solutions appelées voisins. Pour le problème Alignement Multiple des Séquences , cette opération est illustrée par un ensemble de mouvements pour changer les positions des $gaps$ $(' - ')$ dans les séquences de chaque solution. L'opérateur de déménagement utilisé pour construire la structure de voisinage sont comme suit :

Cet opérateur choisit aléatoirement un Gap d'une séquence i et déplacez-le à une position j dans la direction gauche/droite. La nouvelle position j peut être $j + 1$ ou $j - 1$ (droite/gauche) dans la séquence i, Il doit être $j + 1$ ou $j - 1$ est égal à un caractère. Les paramètres i , j et la direction (droite/gauche) dans les règles de déplacement peuvent être déterminés de façon aléatoire.

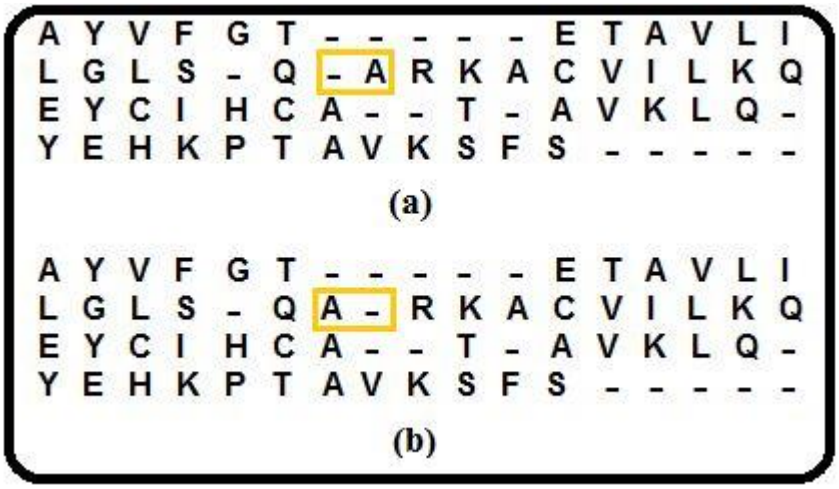

Figure IV.2 : *(a) détection d'un gap et caractère, (b) le changement.*

Nous utilisons une matrice binaire pour détecter le Gap, 1 pour le Gap et 0 pour le caractère.

Séq 1	0	0	0	1	1	0
Séq 2	0	0	1	0	0	0

Figure IV.3 : *Matrice binaire pour détecter le gap.*

3.2.4. La liste tabou

d'où le nom de la méta-heuristique tabou. La taille de la liste tabou est limitée, ce qui empêche l'enregistrement de toutes les solutions rencontrées. C'est la raison pour laquelle la liste tabou procède comme une pile $FIFO$, où la plus ancienne solution sera écartée pour laisser place à la dernière solution rencontrée.

Nous avons utilisé une approche statique pour crée la liste tabou et déterminer la taille de liste tabou. Par default, le choix de la taille de la liste tabou est 20.

3.2.5. Critère d'aspiration

permettant de sortir du blocage causé par la ressemblance des attributs des solutions tabou. Le critère d'aspiration permet d'omettre le statut tabou sur une solution lorsque certaines circonstances sont respectées.

Nous avons utilisé le critère d'aspiration classique consiste à autoriser une solution tabou si celle-ci est l'une des meilleures solutions rencontrées depuis le démarrage de la recherche.

3.2.6. Critères de résiliation

La meilleure solution dans la liste tabou est renvoyée, après une certaine nombre d'itération peut être d'entrée.

4. Les données de test

Afin de validé notre approche, nous avons effectué des tests a partir de différents références de $BAliBASE$. Le nombre de séquences contenues dans chaque ensemble et les tailles des ces séquences sont très variés ce qui permet de bien tester notre approche. Nous présentons dans le tableau suivant les données de test utilisées en donnons le nom de chaque ensemble, le nombre de séquences, la taille de la plus petite séquences, la taille de la plus grande.

L'ensemble des Références	Nombre de séquence	Taille de la plus petite séquence	Taille de la plus grande séquence
BB11001	4	83	91
BBS11013	5	49	57
BBS11022	4	58	69
BBS11027	7	172	217
BBS12014	9	49	52
BBS12039	11	55	81
BBS20002	20	42	49
BBS30006	18	81	121

Tableaux IV.1 : *Les données de test utilisées.*

5. Critères d'évaluation

Pour évaluer la performances de l'algorithme on a fait le test selon deux critères, le taux d'amélioration entre le score initiale et le score final et facteur de temps d'exécution de chaque algorithme. Notons que le taux d'amélioration TA est donné par :

$$TA = \frac{Score\ Initial - Score\ Final}{Score\ Initial} \times 100$$

6. Résultats obtenus par l'algorithme recherche tabou

6.1. Par la matrice PAM250

Référance				Résultat			
Nom	Number des Séq	Séq Min	Séq Max	Score Initial	Score Final	T'aux d'amélioration	Temps d'execution (S)
BB11001	4	83	91	-1394	-603	56,74%	51
BBS11013	5	49	57	-2254	-1010	55,19%	29
BBS11022	4	58	69	-1884	-675	64,17%	24
BBS11027	7	172	217	-13714	-9894	27,85%	124
BBS12014	9	49	52	-7727	-2597	66,39%	35
BBS12039	11	55	81	-19545	-13282	32,04%	55
BBS20002	20	42	49	-36028	-20791	42,29%	72
BBS30006	18	81	121	-59195	-46191	21,96%	134

Tableaux IV.2 : *Les résultat obtenus pour (TL=20,nbr_itération=100) .*

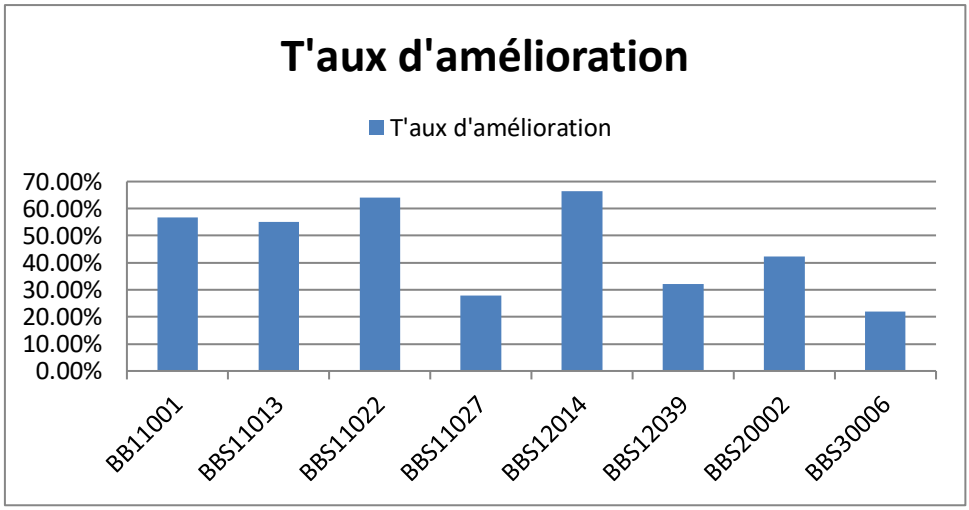

Figure IV.4 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=100).*

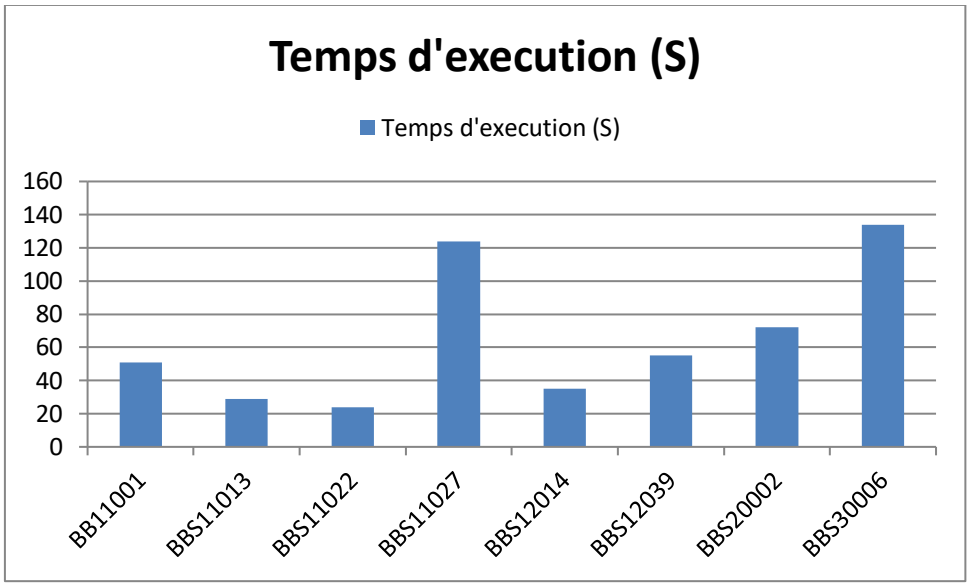

Figure IV.5 : *Temps d'exécution pour (TL=20,nbr_itération=100).*

Référance				Résultat			
Nom	Number des Séq	Séq Min	Séq Max	Score Initial	Score Final	T'aux d'amélioration	Temps d'execution (S)
BB11001	4	83	91	-1418	-410	71,08%	40
BBS11013	5	49	57	-2296	-957	58,31%	31
BBS11022	4	58	69	-1755	-558	68,20%	34
BBS11027	7	172	217	-13633	-8656	36,50%	204
BBS12014	9	49	52	-7505	-1668	77,77%	50
BBS12039	11	55	81	-19149	-11501	39,93%	91
BBS20002	20	42	49	-32276	-14731	54,35%	105
BBS30006	18	81	121	-58977	-40684	31,01%	251

Tableaux IV.3 : *Les résultat obtenus pour (TL=20,nbr_itération=200) .*

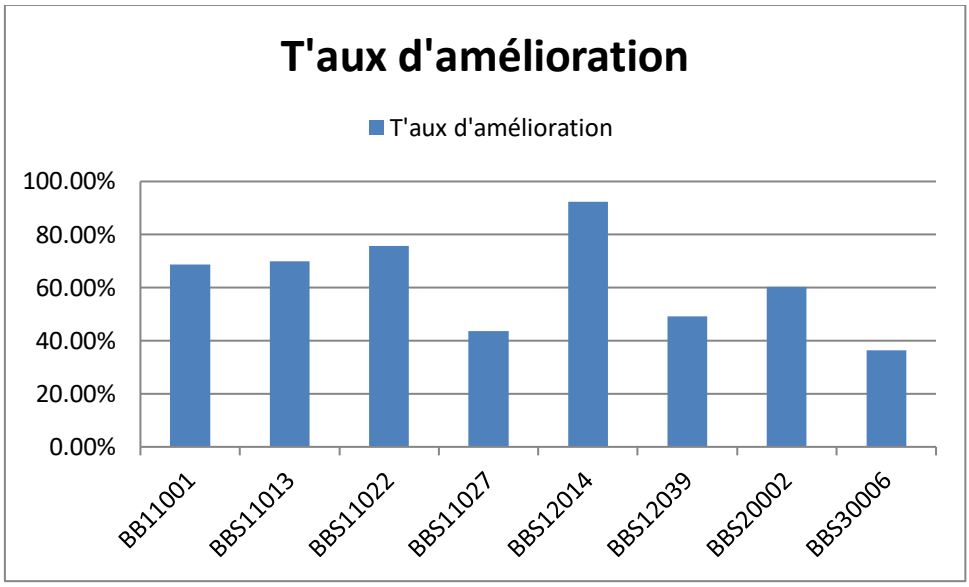

Figure IV.6 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=200).*

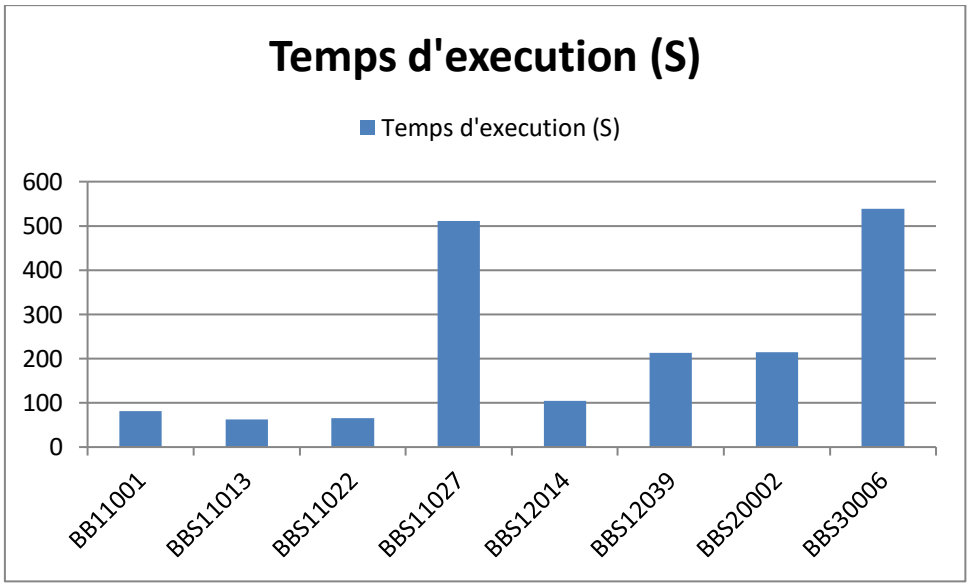

Figure IV.7 : *Temps d'exécution pour (TL=20,nbr_itération=200).*

Référance				Résultat			
Nom	Number des Séq	Séq Min	Séq Max	Score Initial	Score Final	T'aux d'amélioration	Temps d'execution (S)
BB11001	4	83	91	-1309	-409	68,75%	82
BBS11013	5	49	57	-2208	-662	70,01%	63
BBS11022	4	58	69	-1783	-432	75,77%	66
BBS11027	7	172	217	-12703	-7168	43,57%	511
BBS12014	9	49	52	-7285	-557	92,35%	105
BBS12039	11	55	81	-20522	-10450	49,07%	213
BBS20002	20	42	49	-34507	-13717	60,24%	215
BBS30006	18	81	121	-57933	-36862	36,37%	539

Tableaux IV.4 : *Les résultat obtenus pour (TL=20,nbr_itération=500).*

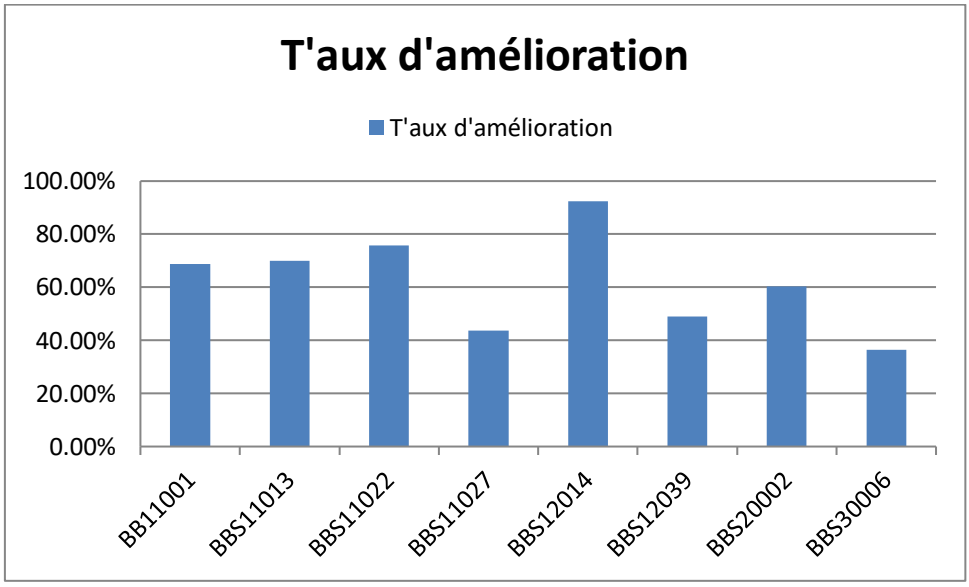

Figure IV.8 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=500).*

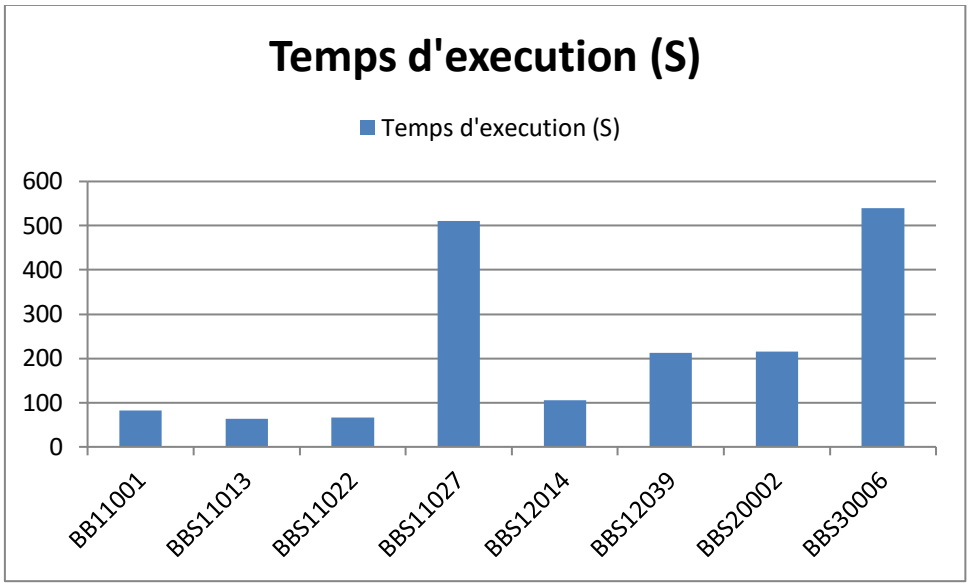

Figure IV.9 : *Temps d'exécution pour (TL=20,nbr_itération=500).*

6.2. Par la matrice BLOSUM62

Référance				Résultat			
Nom	Number des Séq	Séq Min	Séq Max	Score Initial	Score Final	T'aux d'amélioration	Temps d'execution (S)
BB11001	4	83	91	-1376	-687	50,07%	24
BBS11013	5	49	57	-2401	-1076	55,18%	20
BBS11022	4	58	69	-1750	-695	60,28%	27
BBS11027	7	172	217	-14032	-9832	29,93%	133
BBS12014	9	49	52	-7142	-3539	50,44%	36
BBS12039	11	55	81	-19724	-13763	30,22%	55
BBS20002	20	42	49	-35132	-23954	31,81%	52
BBS30006	18	81	121	-60848	-44131	27,47%	192

Tableaux IV.5 : *Les résultat obtenus pour (TL=20,nbr_itération=100).*

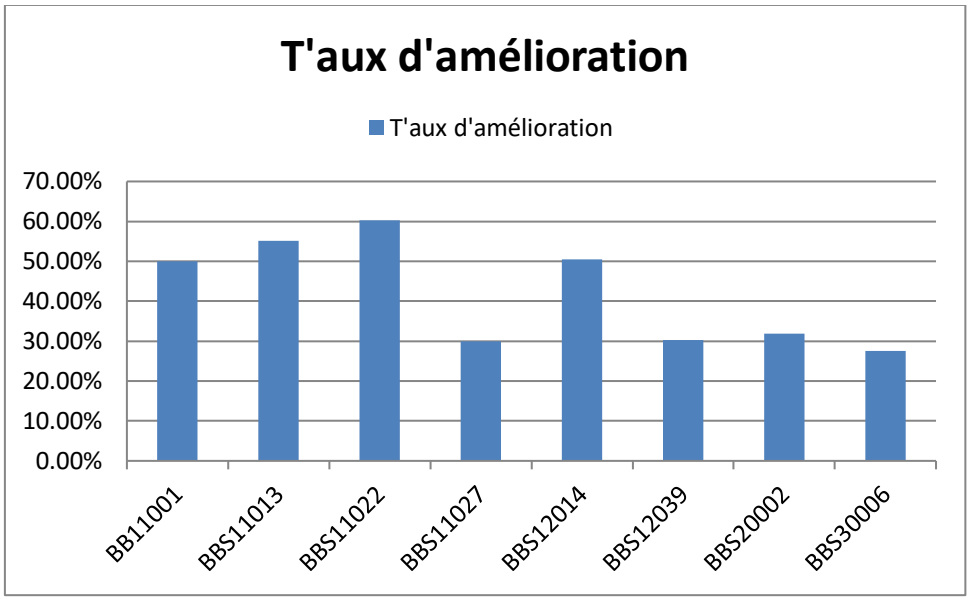

Figure IV.10 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=100).*

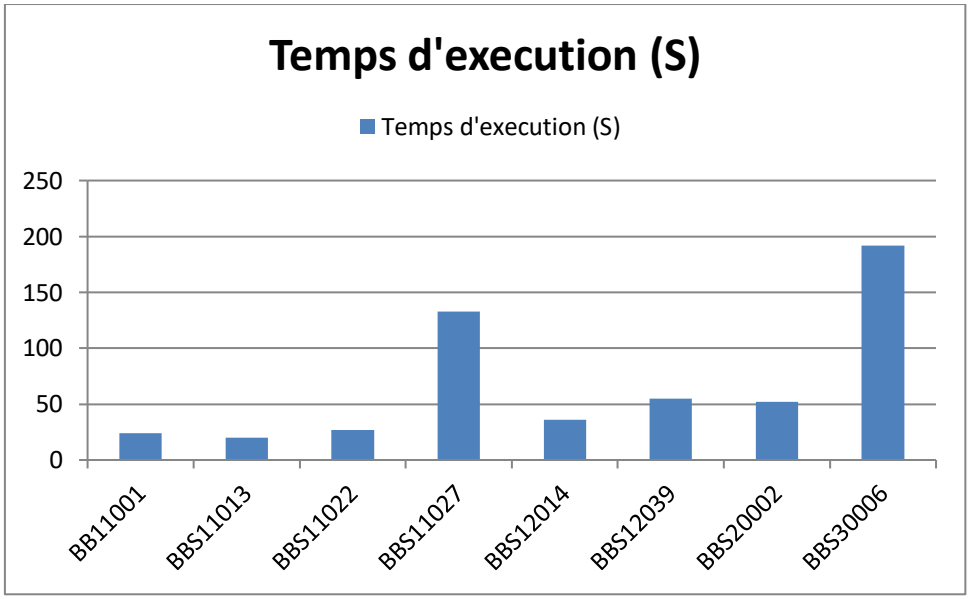

Figure IV.11 : *Temps d'exécution pour (TL=20,nbr_itération=100).*

.

Référance				Résultat			
Nom	Number des Séq	Séq Min	Séq Max	Score Initial	Score Final	T'aux d'amélioration	Temps d'execution (S)
BB11001	4	83	91	-1493	-379	74,61%	40
BBS11013	5	49	57	-2264	-819	63,82%	31
BBS11022	4	58	69	-1798	-662	63,18%	33
BBS11027	7	172	217	-13568	-8774	35,33%	221
BBS12014	9	49	52	-8096	-2332	71,19%	47
BBS12039	11	55	81	-19759	-11843	40,06%	94
BBS20002	20	42	49	-35716	-20562	42,42%	96
BBS30006	18	81	121	-58258	-41991	27,92%	232

Tableaux IV.6 : *Les résultat obtenus pour (TL=20,nbr_itération=200).*

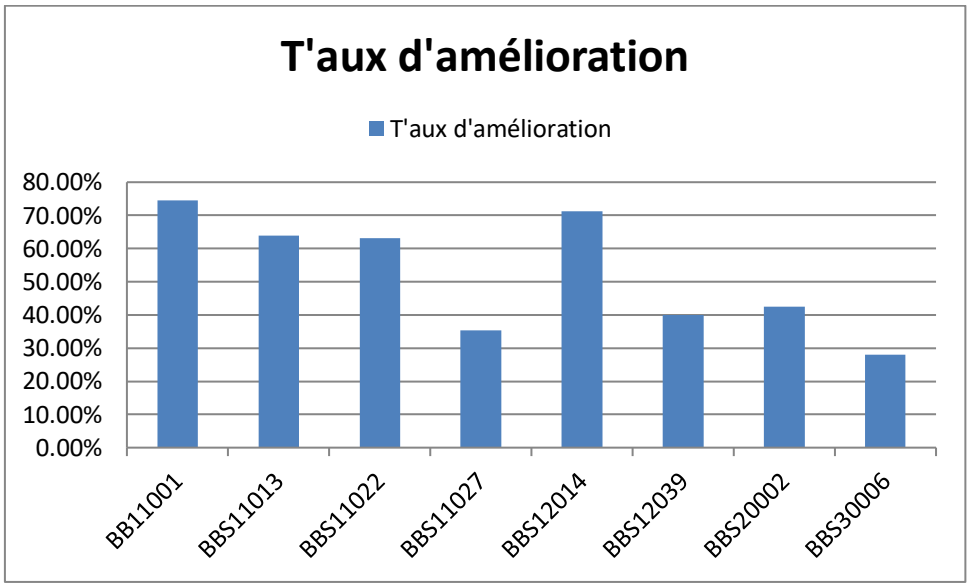

Figure IV.12 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=200).*

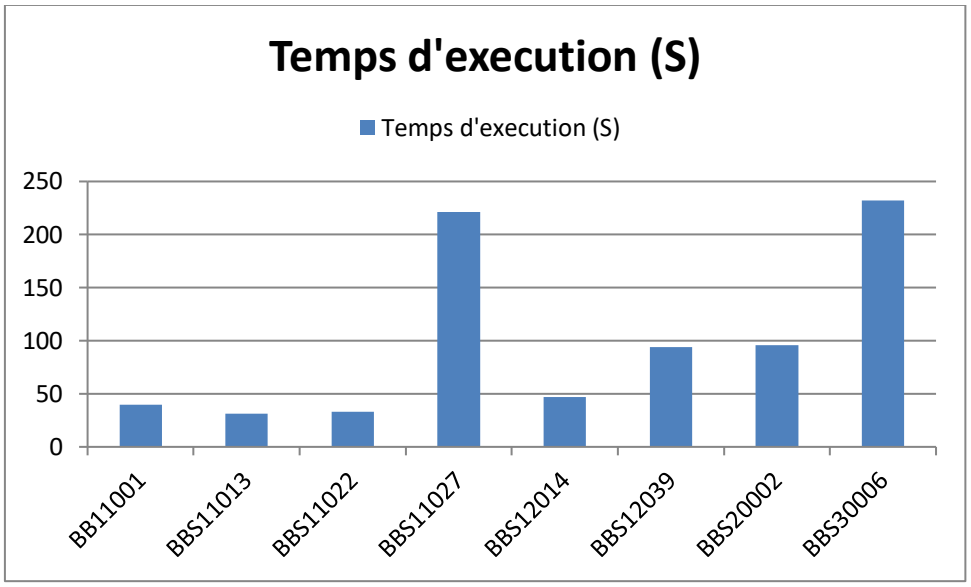

Figure IV.13 : *Temps d'exécution pour (TL=20,nbr_itération=200).*

Référance				Résultat			
Nom	Number des Séq	Séq Min	Séq Max	Score Initial	Score Final	T'aux d'amélioration	Temps d'execution (S)
BB11001	4	83	91	-1493	-379	74,61%	40
BBS11013	5	49	57	-2264	-819	63,82%	31
BBS11022	4	58	69	-1798	-662	63,18%	33
BBS11027	7	172	217	-13568	-8774	35,33%	221
BBS12014	9	49	52	-8096	-2332	71,19%	47
BBS12039	11	55	81	-19835	-9984	49,66%	46
BBS20002	20	42	49	-35716	-20562	42,42%	96
BBS30006	18	81	121	-58258	-41991	27,92%	232

Tableaux IV.7 : *Les résultat obtenus pour (TL=20,nbr_itération=500).*

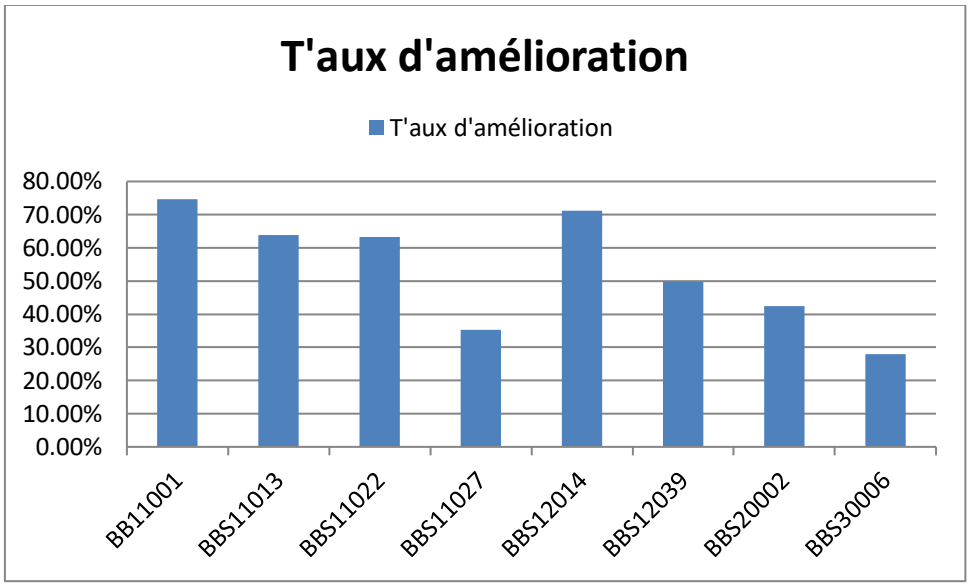

Figure IV.14 : *T'aux d'amélioration de score pour (TL=20,nbr_itération=500).*

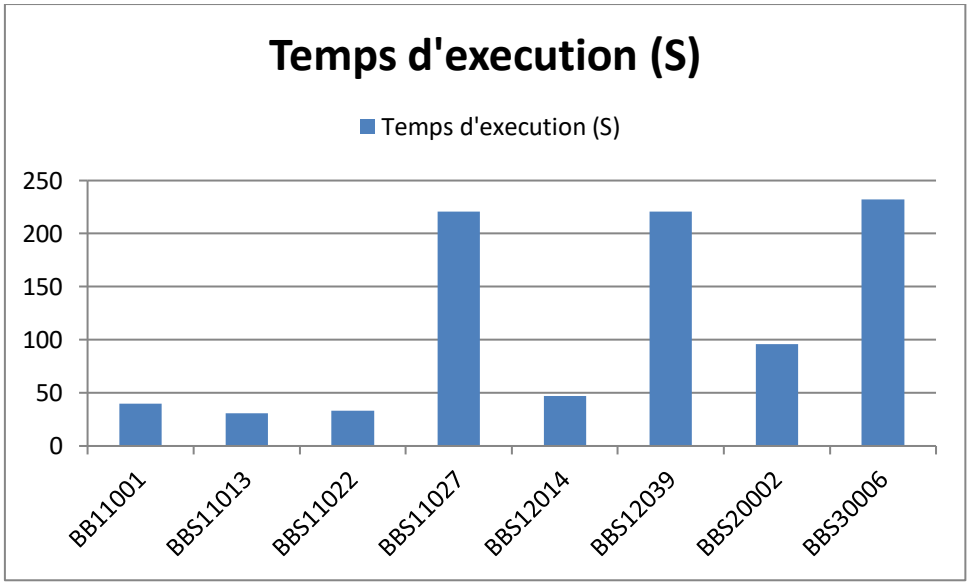

Figure IV.15 : *Temps d'exécution pour (TL=20,nbr_itération=500).*

6.3. Analyse des résultat pour la recherche tabou

Nous allons comparer les résultats précédents, pour montrer les améliorations.

$$Average = \frac{\sum_{i=1}^{n} S}{\sum_{i=1}^{n} N}$$

- S : La résultat de la référence,
- N : Nombre de la référence.

N	Réf	PAM250			BLOSUM62		
		iter = 100	iter = 200	iter = 500	iter = 100	iter = 200	iter = 500
1	BB11001	56,74%	71,08%	68,75%	50,07%	74,61%	74,61%
2	BBS11013	55,19%	58,31%	70,01%	55,18%	63,82%	63,82%
3	BBS11022	64,17%	68,20%	75,77%	60,28%	63,18%	63,18%
4	BBS11027	27,85%	36,50%	43,57%	29,93%	35,33%	35,33%
5	BBS12014	66,39%	77,77%	92,35%	50,44%	71,19%	71,19%
6	BBS12039	32,04%	39,93%	49,07%	30,22%	40,06%	49,66%
7	BBS20002	42,29%	54,35%	60,24%	31,81%	42,42%	42,42%
8	BBS30006	21,96%	31,01%	36,37%	27,47%	27,92%	27,92%
	Average	45,83%	54,64%	62,02%	41,93%	52,32%	53,52%

Tableaux IV.8 : *Tout les résultat obtenus par (PAM250 et BLOSUM62).*

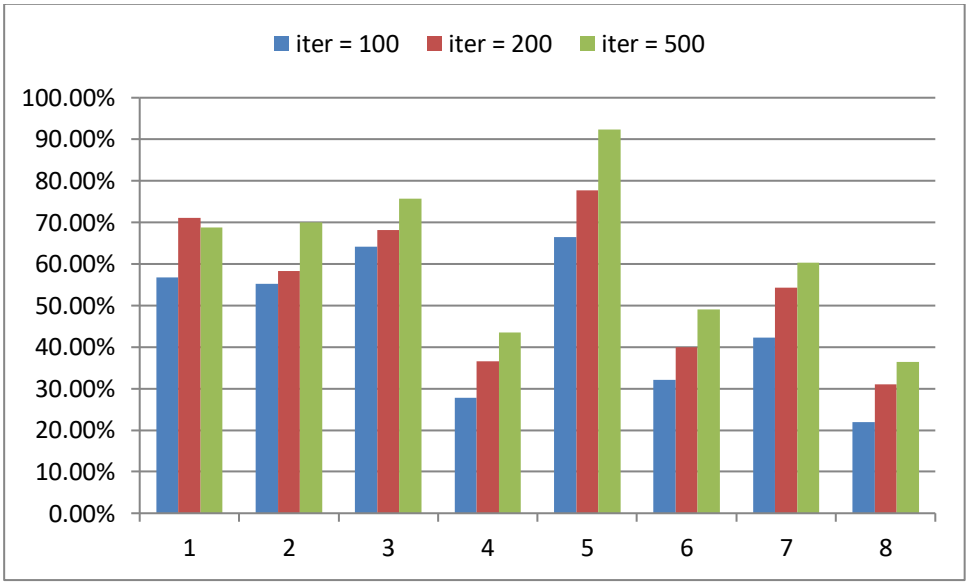

Figure IV.16 : *Taux d'amélioration de score global pour chaque itération lorsque en utilisant la matrice PAM250 .*

L'analyse des résultats présentés dans les tables (IV.2 et IV.3 et IV.4 et IV.8) et la figure (IV.16) montrent clairement que la matrice PAM250 améliore grandement la solution initiale, et cette amélioration est largement lié au nombre d'itérations.

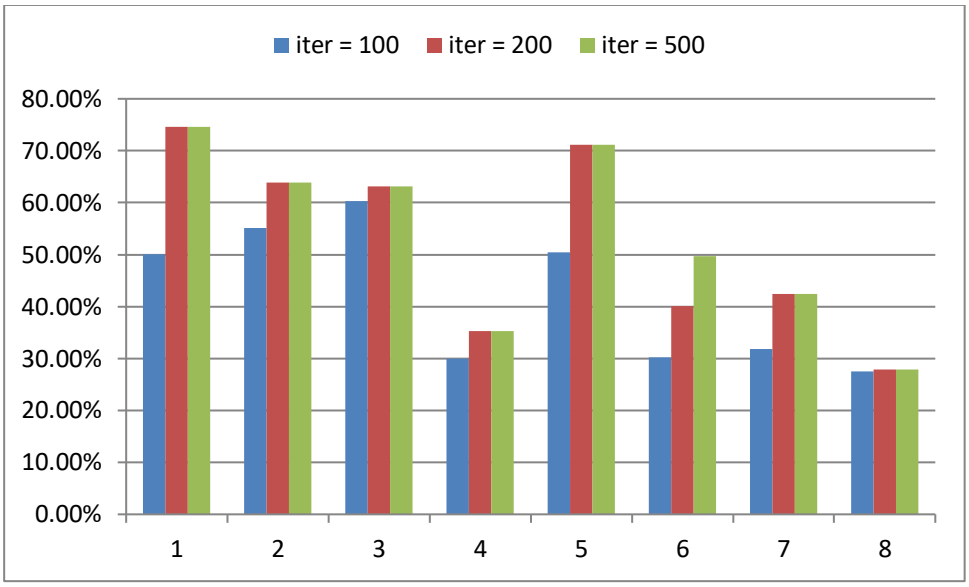

Figure IV.17 : *Taux d'amélioration de score global pour chaque itération lorsque en utilisant la matrice BLOSUM62 .*

L'analyse des résultats présentés dans les tables (IV.5 et IV.6 et IV.7 et IV.8) et la figure (IV.17) montrent clairement que la matrice BLOSUM62 améliore la solution initiale, et après le nombre d'itération est supérieur ou égal à 200, on note que les résultats sont stables.

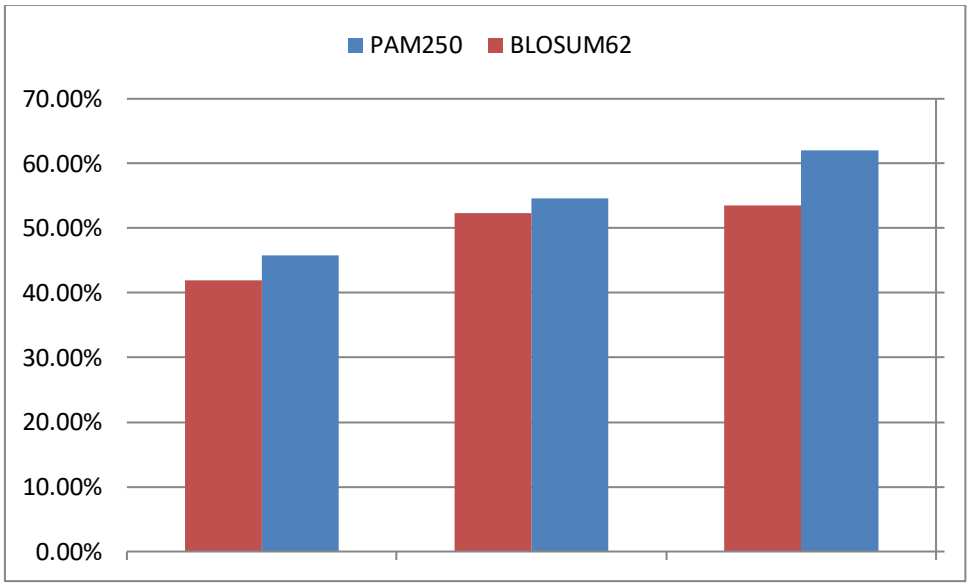

Figure IV.18 : *L'histogramme des moyen des Taux d'amélioration de score.*

D'après la figure (IV.18) la matrice PAM250 marque un meilleur taux d'amélioration de score que BLOSUM62.

7. Conclusion

Dans ce chapitre nous venons de présenter une approche pour la résolution du problème d'alignement multiple des séquences, est appelé l'algorithme recherche tabou pour l'alignement multiple de séquence.

Nous avons utilisé la fonction somme des paires (SP), et les matrice de substitution (PAM250 et BLOSUM62), pour faire une comparaison avec les matrices de substitutions d'après les moyen des Taux d'amélioration de score.

Conclusion générale

Conclusion générale

Dans le cadre de ce travail de master, nous avons traité un problème très important en bioinformatique celui de l'alignement multiple de séquences (MSA) par l'algorithme recherche tabou. Ainsi que l'algorithme recherche tabou c'est un approche itératif, il nécessitent en général de tempe de calculs très important. La fonction de fitness de l'exploration des solution voisinage utilisée par algorithme recherche tabou est la Somme des Paires (SP). Nous avons utilisé deux déférents matrices des substitutions, la matrice BLOSUM62 et PAM250, et le calcul du cout des brèches est effectué suivant le modèle de cout affine avec comme paramètre par défaut le cout d'ouverture d'une brèche est $gop = -10$, et le cout de l'extension de la brèche $gep = -1$. Avec cette matrice de substitution qui choisi et ce modèle de calcul de cout des brèches, la fonction Somme des Paire donne en générale une bonne évaluation des alignements multiples.

Nous avons utilisé différents des ensembles de référence de BAliBASE (version3.0). Ensuite, nous avons comparé les résultats obtenus par l'algorithme recherche tabou avec les matrices de substitutions, PAM250 et BLOSUM62 d'après les résultat obtenu après certain nombre d'itération, nos avons remarqué que la matrice PAM250 donnent un meilleur taux d'amélioration de score que la matrice BLOSUM62.

Bibliographie

[1] Damiens IMBS et MOHAMED SAYED HASSAN, Bioinformatique travail d'étude , Univ de nice sophia antipolis.

[2] Jean-Claude Callen, Avec la collaboration de Roland Perasso," biologie cellulaire : des molécules aux organismes", Edition,2. Publisher, Dunod, 2005. ISBN, 2100492365, 9782100492367.

[3] Alberts,02 A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, "Molecular Biology of the Cell", Garland Science, 4ème édition 2002.

[4] Nadira Benlahrache, " Optimisation Multi-Objectif Pour l'Alignement Multiple de Séquences ".

[5] Edgar, 04 R.C Edgar, " MUSCLE: multiple sequence alignment with high accurac and highthroughput ", Nucleic Acids Res . Vol. 32, No. 5, pp. 1792-1797, 2004.

[6] S.F. Altshul, R.J. Carroll and D.J. Lipman, "weights for data related by a tree", J. Mol. Biol. Vol. 207, pp. 647-653, 1989.

[7] Vincent Derrien ," Heuristiques pour la résolution du problème d'alignement multiple ", Thése de doctorat , N°d'ordre 885, 2008.

[8] Guillaume Lecointre, MNHN , Relations de parenté entre les êtres vivants , http://acces.ens-lyon.fr/biotic/evolut/parente/html/arbphyl.htm.

[9] Rong etHansen, Z. Rong and E.A. Hansen. "K-Group A* for Multiple Sequence Alignment with Quasi Natural Gap Costs", 16th IEEE International Conference on Tools with Artificial Intelligence. Boca Raton, FL. November, 2004.

[10] Vincent Derrien, Jean-Michel Richer,Jin-Kao Hao." Plasma, un nouvel algorithme progressif pour l'alignement multiple de séquences ". LERIA - Université d'Angers, 2 Bd Lavoisier, 49045 Angers, France.

[11] NCBI , http://www.ncbi.nlm.nih.gov, consulté le 17/03/ 2018.

[12] V.I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and Reversals". Soviet Physics Doklady, 10:707–710,1966.

[13] S.MESHOUL, "Approche quantique évolutionnaire pour l'alignement multiple de séquences en bioinformatique" , Magistère, Université Mentouri de Constantine, 2005.

[14] M O Dayhoff,R.M.Schwartz and B C Orcutt, "A model of evolutionary change in proteins ", Atlas Protein Seq,Struct,vol.6 pp.345-362.1978.

[15] S Henikoff and J Henikoff."Amino acid substitution matrices from protein blocks",Proceedings of the National Academy of Sciences,No.89,pp.915-919, 1992.

[16] Batzoglou, 04. S.Batzoglou , “Sequence Alignment’ I: CS262 Winter 2004: Lecture II, 2004.

[17] Wallace, 04 I. M. Wallace, O. O’Sullivan, D. G. Higgins and C. Notredame. " M-Coffee: combining multiple sequence alignment methods with T-Coffee", Nucleic Acids Res. , 2006, Vol. 34, No. 6pp.1692–1699.

[18] Smith et Waterman, T. Smith and M. Waterman, "Identification of common molecular subsequence".J. Mol. Biol . Vol. 147, pp. 195-197. 1981.

[19] Stoye et autres, J. Stoye, V, Moulton, and A. W. Dress, "DCA, an efficient implementation of the divide and conquer approach to simultaneous multiple sequence alignment", Comput. Appl. Biosc., Vol. 13, No. 6, pp. 625-631, 1997.

[20] D.F. Feng and R.F Doolittle. "Progressive sequence alignment as a prerequisite to correct phylogenetic trees". J. Mol. Evol ., Vol. 25, pp.351-360, 1987.

[21] S .F. Altschul, "Amino acid substitution matrices from an information theoretic perspective". Journal of Molecular Biology, 219: 555–565, 1991.

[22] H.Carillo DJ Lipman,"The multiple sequence alignment problem in biology", SIAM J. Appl. Math 1988.

[23] W.R.Taylor,"Multiple sequence alignment by a pairwise algorithm". Comput Appl Biosci, Vol. 3, pp. 81-7. 1987.

[24] G. J.Barton, and M. J. Sternberg, "A strategy for the rapid multiple alignment of protein sequences. Confidence levels from tertiary structure comparisons". J. Mol Biol, Vol. 198, pp. 327-37, 1987.

[25] J.D. Thompson, D.G.Higgins and T.J.Gibson, "CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice", Nucleic Acids Res. Vol. 22 No. 22 pp. 4673-4680, 1994.

[26] C.Notredame, D.G. Higgins and J. Heringa, "T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment ", J. Mol. Biol. Vol. 302, pp. 205- 217, 2000.

[27] R. Chenna H Sugawara T Koike R Lopez TJ Gibson DG. Higgins JD Thompson," Multiple sequence alignment with the clustal series of programs", Nucleic Acids Research,2003.

[28] C Notredame and D.G. Higgins."Saga: Sequence alignment by genetic algorithm". Nucleic Acid Research, Vol.24,1996.

[29] J H Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975.Lecture II, 2004.

[30] https://fr.wikipedia.org/wiki/NetBeans

[31] https://fr.wikipedia.org/wiki/Java_(langage)

[32] Amira Gherboudj," Méthodes de résolution de problèmes difficiles académique", Université de constantine2 , thése de doctorant , 2013.

[33] Bernstein FC, Koetzle TF, Williams GJ, Meyer Jr EF, Brice MD, Rodgers JR, Kennard O, Shimanouchi T, Tasumi M. « The Protein Data Bank: a computer-based archival file for macromolecular structures » *J Mol Biol*. 1977;112:535-542.

[34] F. Glover. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research. Vol . 13, N° 5, pp 533-549, 1986.

[35] F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, Boston.1997.

[36] Notredame,C, "Recent progress in multiple sequence alignment: a survey. Pharmacogenomics", 2002.

[37] M.A. McClure, T. K. Vasi and W.M. Fitch, « Comparative analysis of multiple .

protien sequence alignment methods », Mol. Biol . Evol. Vol. 11 pp. 571-592. 1994.

[38] J.D. Thompson, F. Plewniak, and O. Poch, «BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs". Bioinformatics, Vol. 15, pp. 87 88, 1999.

[39] R.C Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput", Nucleic Acids Res. Vol.32, No. 5, pp. 1792-1797, 2004.

[40] I. Van Walle, I. Lasters, and L. Wyns, "Align-m: a. new algorithm for multiple alignment of highly divergent sequences", Oxford University Press, 2004.

[41] J. Stoye, V, Moulton, and A. W. Dress, « DCA, an efficient implementation of the divide and conquer approach to simultaneous multiple sequence alignment", Comput. Appl. Biosc., Vol. 13, No. 6, pp. 625-631, 1997.

الملخص :

تعتبر السلاسل البيولوجية المتعددة (MSA) من العمليات الأساسية للعديد من التطبيقات في مجال البيومعلوماتية الذي يسعى الى المعالجة الآلية للمعلومة البيولوجية.

في هذا العمل المخصص لمذكرة نهاية الدراسة ، قمنا بتطبيق خوارزمية البحث تابو $(Tabu\ Search)$ وذلك من أجل حل مشكل السلاسل البيولوجية المتعددة، بحيث تم تطبيق كل ضوابط الخوارزمية على مجموعة من السلاسل البيولوجية، ثم بعد ذلك تحصلنا على نتائج محسنة بعد العديد من التكرارات.

الكلمات المفتاحية : البيومعلوماتية، السلاسل البيولوجية المتعددة، خوارزمية البحث تابو.

Abstract :

The multiple sequence alignment are considered a fondamental part of the processes of a multitude of applications in the bioinformatic field whose function is the automatic processing of biological information.

In this study, dedicated to the fulfillment of the dissertation, we have implemented an algorithm $tabu\ search$ in order to solve the problem of the multiple sequence alignment, in a way that all algorithm regulations were applied on a set of alignments. Thus, we have obtained improved results after many number of repetition.

Keywords : Bioinformatic, Multiple Sequence Alignment**,** Algorithm Tabu Search.

Résumé :

Les alignement multiple de séquences sont considérés comme une partie fondamentale des processus d'une multitude d'applications dans le domaine bioinformatique, qui sert à traiter automatiquement l'information biologique.

Dans cette étude destinée à la mémoire de fin d'étude, nous avons implémenté un algorithme $recherche\ tabou$ pour trouver une solution au problème des alignements multiple de séquences , de sorte que tous les règlement de l'algorithme ont été appliqués sur un ensemble d'alignements. Ainsi, on a obtenu des résultats améliorés après maintes nombre des répétitions.

Mot clé : Bioinformatique, Alignements Multiple de Séquence, Recherche Tabou.