REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

N° d'ordre : .....

UNIVERSITE DE M'SILA

FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE

Département d'Informatique

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique
Filière : Informatique
Spécialité :Réseaux

Par : SERIBLI Houssam Eddine

SUJET

Développement et Implémentation d'un Solveur Bio-inspiré pour l'Alignement de Séquences Biologiques

Soutenu publiquement le : 14 / 06 /2015 devant le jury composé de :

Dr. LAMICHE Chaabane	**Université de M'sila**	**Rapporteur**
……...	**Université de M'sila**	**Président**
……	**Université de M'sila**	**Examinateur**
……	**Université de M'sila**	**Examinateur**

Promotion : 2014 /2015


بسم الله الرحمن الرحيم

Remerciements

*Nous tenons à exprimer nos vifs remerciements à **ALLAH** qui nous a donné la volonté et la patience afin de finir notre étude malgré les difficultés rencontrées.*

Nous tenons à remercier par ce travail tous ceux qui nous ont aidés de loin ou de prés à réaliser ce mémoire de fin d'étude :

*Nous remercions **Dr.LAMICHE Chaabane** qui a bien voulu accepter l'encadrement de ce mémoire et nous a apporté des conseils précieux durant toutes les étapes de ce travail.*

Nous souhaitons également remercier les membres du jury pour nous avoir fait l'honneur d'accepter de juger et d'évaluer notre travail.

Nous remercions tous nos enseignants pour toutes les connaissances Qu'ils nous ont inculquées tout au Lang des deux années.

Table des Matières

CHAPITRE III : Optimisation par Métaheuristiques à Population

CHAPITRE IV: Réalisation et expérimentation

Liste Des Figures

Liste des tableaux

Introduction générale

Introduction générale

La bioinformatique est un domaine pluridisciplinaire où l'informatique joue un rôle prépondérant. C'est une science qui conceptualise la biologie en termes de molécules et applique des " techniques d'informatiques" pour modéliser, analyser, comparer et simuler l'information biologique incluant séquences, structures, fonctions et phylogénie. L'alignement multiple de séquences ou MSA (pour **M**ultiple **S**equence **A**lignment) est un problème fondamental en biologie moléculaire et représente une tâche de base pour beaucoup d'applications en bioinformatique. Il vise à apparier au sens biologique plusieurs séquences nucléiques et protéiques. MSA est le moyen utilisé par les biologistes pour analyser des séquences d'ADN (nucléiques) ou de protéines (protéiques) afin de déterminer leur degré d'homologie ou de divergence. La recherche d'un alignement de bonne qualité implique souvent l'exploration d'espaces de recherche très vastes et dont la taille devient de plus en plus critique avec le nombre et les tailles des séquences à aligner. Cependant trouver un alignement multiple a été démontré un problème NP-complet, MSA ne peut être résolu par une méthode exacte que pour des séquences de petites tailles et dont le nombre est réduit induisant des espaces de tailles réduites.

Le recours aux méthodes itératives pour gérer la complexité combinatoire du problème est désiré. Leur principe de base consiste à produire un alignement initial et à le raffiner itérativement de manière déterministe ou stochastique. Dans notre travail nous proposons de concevoir un solveur bio-inspiré basé sur deux algorithmes métaheuristiques à population telle que : La méthode d'optimisation par essaim de particules et les algorithmes génétiques pour la résolution du problème MSA. Nous avons organisé notre mémoire comme suit :

le chapitre 1 introduit le domaine de la biologie moléculaire et la bioinformatique. Le chapitre 2 fournit une description du problème de l'alignement multiple de séquences et les déférentes approches de résolution. Le chapitre 3 est dédié à la description des algorithmes utilisés et à l'adaptation de chaque algorithme au problème MSA. Le chapitre 4 sera entièrement consacré la partie réalisation et aux résultats expérimentaux. Le mémoire s'achèvera par une conclusion et des perspectives.

CHAPITRE I

Introduction à la BioInformatique

1. Introduction

La bioinformatique est l'utilisation des technologies de l'information dans le domaine de la biologie moléculaire. Bioinformatique implique maintenant la création et le développement de bases de données, des algorithmes, des techniques informatiques et statistiques et de la théorie pour résoudre les problèmes formels et pratiques découlant de la gestion et l'analyse des données biologiques.

Dans ce chapitre, on propose une présentation générale sur la bioinformatique et notion primaire sur La Biologie Moléculaire et les domaines de recherche pour les bioinformaticien.

2. La Bioinformatique

Définition : bio - informatique: science qui conceptualise la biologie en termes de molécules (dans le sens de la chimie-physique) et applique des " techniques d'informatiques " pour comprendre et organiser l'information liée à ces molécules, sur une grande échelle. En bref, la bioinformatique est un système intégré de gestion pour la biologie moléculaire et a beaucoup d'applications pratiques [12].

Le mot « bioinformatique » découle donc de l'analyse par ordinateur des données biologiques. Ces données représentent l'information stockée dans le code génétique, mais également des résultats expérimentaux de diverses sources et des statistiques, … etc.

La bioinformatique est une science récente qui évolue rapidement et qui est fortement interdisciplinaire, elle conjugue plusieurs sciences telles que la biologie moléculaire, l'informatique, et les mathématiques (statistiques)… etc. Le but de la recherche dans la bioinformatique est l'organisation et l'extraction des données, la mise en application des algorithmes complexes et le développement des outils de visualisation afin d'atteindre une compréhension exhaustive et une exploitation des informations contenues dans les séquences d'un génome.

L'histoire du calcul dans la biologie moléculaire n'est pas récente mais date des années 20 où les scientifiques pensaient déjà à établir des lois biologiques par induction. Cependant, seulement le développement des ordinateurs puissants, et la disponibilité des données expérimentales qui peuvent être aisément traitées par calcul (par exemple, les séquences d'ADN ou d'acide aminé et des structures tridimensionnelles des protéines) ont lancé la bioinformatique comme un domaine indépendant. Aujourd'hui, les applications pratiques de la bioinformatique sont aisément disponibles sur le Web, et sont largement répandues dans la recherche biologique et médicale.

Le rapport entre l'informatique et la biologie moléculaire est normal pour plusieurs raisons. D'abord, le taux phénoménal de données biologiques produites fournit des défis: des quantités massives de données doivent être stockées, analysées, et doivent être rendues accessibles [29]

En second lieu, les données sont souvent exprimées comme des formules statistiques, et par conséquent le calcul, est nécessaire. Ceci s'applique en particulier aux informations sur la construction des protéines et de l'organisation temporelle et spatiale de leur expression dans la cellule, troisièmement il y a une analogie forte entre la séquence d'ADN et un programme machine ; une séquence d'ADN représente une machine de Turing [12].

Mais l'analyse bioinformatique est également appliquée à de diverses autres données, par exemple arbres phylogéniques, les réseaux métaboliques, et les statistiques [32]. Une myriade de techniques sont employées, y compris l'alignement de séquences primaires, alignement de la structure 3D de la protéine, la construction d'arbre phylogénétique, la prédiction et la classification de la structure de protéine, la prédiction de la structure d'ARN, la prédiction de la fonction de protéine, et l'expression de données groupées. Le développement algorithmique occupe une partie importante dans la bioinformatique. Des algorithmes complexes ont été spécifiquement développés pour l'analyse et l'accès aux données biologiques, par exemple : l'algorithme de programmation dynamique pour l'alignement de séquence, les programmes d'interrogations des bases de données biologiques tels que : BLAST, FASTA [2],

La Bioinformatique a un grand impact sur la recherche biologique. Les projets de recherche géants tels que le projet humain de génome, seraient sans signification sans la composante bioinformatique. Une fois que les données brutes sont disponibles, des hypothèses peuvent être formulées et évaluées *in silico* [32]. De cette manière, les expériences menées par ordinateur peuvent répondre aux questions biologiques qui ne peuvent pas être abordées par des approches traditionnelles. Ceci a mené à la fondation des laboratoires de recherche dédiés seulement à la bioinformatique.

Cette science peut être définie sur trois axes : Acquisition et organisation des données biologiques, conception des logiciels pour l'analyse, la comparaison et la modélisation des données et le dernier axe est l'analyse des résultats produits par les logiciels. Les thèmes traités par la bioinformatique sont :

- Modélisation et représentation de la connaissance en base de Données.
- Méthodes de comparaison de chaîne de caractères comme recherche mots et des textes.

- Algorithmes et techniques d'alignement de séquences et alignement multiple de séquences.
- Identification de motif et modèle pour des séquences multiples
- Analyse et interprétation : Techniques de data-mining (la fouille des données).
- Représentation graphique des surfaces et des volumes, et comparaison structurelle 3D
- Simulations moléculaires.
- Les analyses statistiques afin de fournir une mesure objective pour la signification des résultats.
- Réalisation des interfaces web pour faciliter l'accès aux banques de données à travers le monde.

Afin de pouvoir comprendre et assimiler les thèmes traités par la bioinformatique, il devient nécessaire de présenter quelques notions de la biologie moléculaire mais sans entrer dans des détails métaboliques et physico-chimiques.

3. Génome et Génomique

Le génome d'un organisme vivant constitue l'information génétique qui permet à cet organisme de vivre et d'évoluer. Il contient toute l'information génétique nécessaire au fonctionnement de la cellule et par conséquent de tout l'organisme.

La génomique est la science qui a pour but l'étude exhaustive des génomes, elle constitue actuellement un défi scientifique important sur plusieurs plans. La génomique permet d'étudier l'ensemble des gènes, d'une espèce donnée, leur fonction, leur rôle ainsi que leur répartitions sur les chromosomes et les relations entre eux. Un génome séquencé est un texte formé de quatre lettres (A, C, G, T), il reste un énorme travail de décryptage pour pouvoir interpréter ce texte et d'explorer les structures et les processus moléculaires qui sont fondamentaux à la vie En gros trois tâches restent à réaliser :

- Identification des gènes et leur fonction
- Compréhension des réseaux d'interactions moléculaires.
- Comparer ce génome à celui des autres espaces.

Les intérêts d'un tel travail sont majeurs:

- Évolution des espèces (la théorie de l'évolution)
- Fonctionnement des cellules : comprendre les mécanismes de régulation des gènes.
- Médecine : identifier les gènes qui provoquent des maladies et expliquer les causes des maladies complexes.

- Étude de la propagation des maladies.
- Pharmaceutique : aide à la conception des remèdes et des traitements.
- Écologie : préservation de la faune et de la flore.
- Nutrition : Organismes Génétiquement Modifiés (OGM)

4. La Biologie Moléculaire pour un Bio-informaticien

4.1.L'ADN (Acide Désoxyribonucléique)

La Figure 1.1 montre un schéma abstrait d'une cellule. Il y a un noyau contenant l'ADN. Les protéines sont à l'intérieur de la cellule mais en dehors du noyau. Les acides nucléiques, y compris l'ADN et l'ARN, forment le matériel génétique de tout l'organisme. Ce sont toutes les informations de quoi a besoin un organisme pour fonctionner ainsi que toutes les caractéristiques héréditaires.

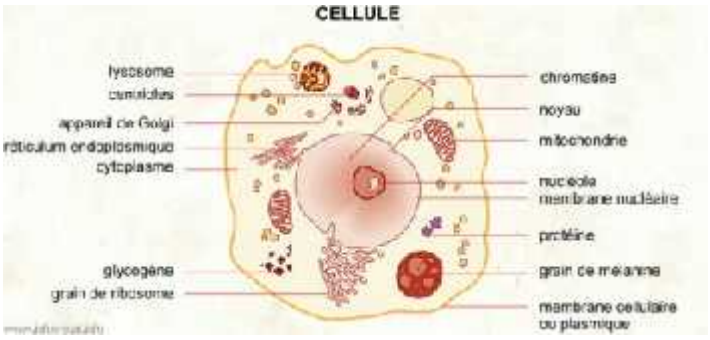


Figure 1.1 : Schéma simplifié d'une cellule

Ce sont des molécules structurées en chaîne, composées des nucléotides

Un nucléotide d'ADN (Figure1.2) a 3 composants: un sucre (désoxyribose), un composant d'acide phosphorique (phosphate), et une base d'azote (un des quatre types : Adénine ou Adénosine (A), Guanine (G), Cytosine (C) et Thymine (T)).

L'ADN peut être en *simple brin* ou *double brin*. Un brin simple (aussi appelé polynucléotide) est un Polymère linéaire (Figure 1.3).

On représente un polynucléotide par une séquence orientée de lettres:

A-T-T-C-A-G-G-C-A-T-T-A-G-C

Les brins de nucléotides peuvent coller ensemble pour former une épine dorsale continue. Ceci donne une forme d'échelle (Figure 1.4).

La forme d'échelle se torde sur elle-même pour donner une forme hélicoïdale (Figure1.5). Cette structure est la célèbre " double hélice ", découverte par Crick et Watson en 1953. Les bases ou nucléotides (A, T, C, G) s'organisent en paires selon une complémentarité exclusive: A-T et G-C. C'est cet appariement qui permet un enroulement quasi-parfait en hélice droite des deux chaînes sucre -phosphate qui portent ces nucléotides [1].

La structure est stabilisée par l'interaction (liaisons d'hydrogène) entre les bases et l'empilement successif des paires de nucléotides (figure 1.6).

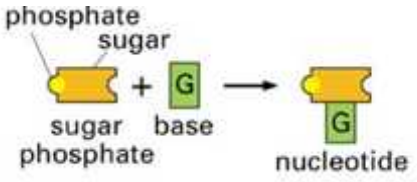


Figure 1.2 : Construction d'un nucléotide ou polynucléotide

Figure 1.3 : Un brin d'ADN

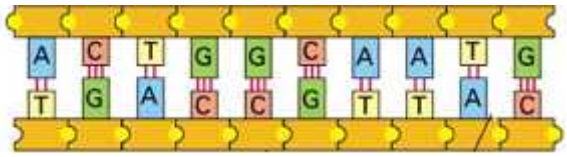


Figure 1.4 Construction du 2ème brin d'ADN (Forme d'échelle)

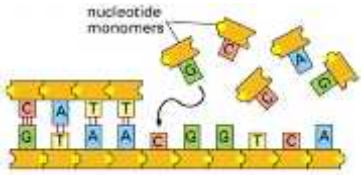


Figure 1.5 : Double brins d'ADN

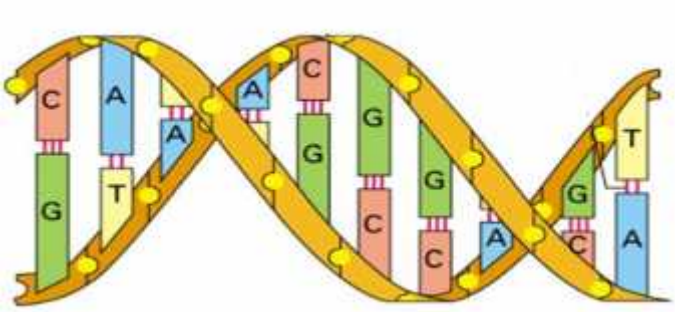


Figure 1.6 : Double brins d'ADN (Forme hélicoïdale)

Dans un brin d'ADN, il y a des segments dits codants (Exons) (figure 1.7) et des segments non codants (Introns). Le premier type qui est l'exon, va participer à la génération d'autres macromolécules (ARNs et par la suite des protéines) contrairement aux introns qui sont sans utilité apparente [1].

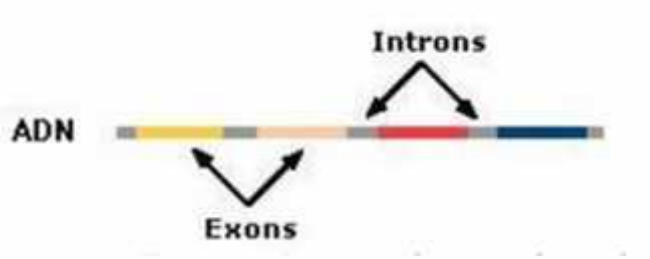


Figure 1.7 : Exons et Introns dans un brin d'ADN

4.2.Les Chromosomes

Les chromosomes sont des éléments du noyau cellulaire en nombre constant, qui déterminent l'hérédité.

Un chromosome est une structure en bâtonnet, constituée de longues chaînes d'ADN, auxquelles sont fixées des protéines. L'ADN de l'homme est divisée en 23 paires de chromosomes contenus dans le noyau de chacune de ces cellules, 22 paires sont communes aux deux sexes.

Les deux chromosomes restants sont les chromosomes sexuels. Chez la femme, ils forment une paire. On les appelle les chromosomes X et l'autre, beaucoup plus court est appelé chromosome Y.

4.3.L'ARN

L'ARN (Acide Ribonucléique) ressemble énormément à l'ADN (figure 1.8) mais il y a des différences telles que :

- ✓ Le sucre de l'ADN (désoxyribose) et celui de l'ARN est le ribose.
- ✓ La Thymine (T) de l'ADN est remplacée par *l'uracile (U)* dans l'ARN
- ✓ l'ARN peut s'apparier avec un autre ARN complémentaire mais les ARNs sont généralement simple brin. Contrairement aux brins de l'ADN qui vont en couple.
- ✓ 3 types d'ARNs ont été identifiés : ARN messager (ARNm), ARN ribosomiques (ARNr) et ARN transfert (ARNt). mais d'autres types ont été découverts ces dernières années.

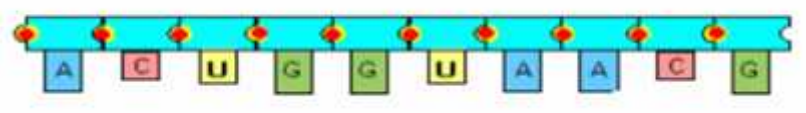


Figure 1.8 : un brin d'ARN

Structures d'un brin ARN

Un brin d'ARN peut avoir plusieurs structures : primaire (Figure 1.9), secondaire (Figure1.10) et tertiaire (Figure 1.11) [4]. Cette définition est valable même pour les protéines à qui on peut attribuer encore une structure quartenaire.

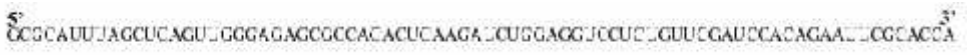


Figure 1.9: La structure primaire d'une séquence d'ARNt de la phénylalanine

Figure 1.10 La structure secondaire d'une séquence d'ARNt du phénylaline

Figure 1.11 : La structure tertiaire d'une séquence d'ARNt du phénylalanine

4.4.Les Protéines

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 [1]:

- ✓ Protéines structurelles: elles sont les bases de construction des divers tissus.
- ✓ Enzymes: elles catalysent les réactions chimiques essentielles qui auraient pris beaucoup de temps pour se produire.
- ✓ Transporteuses: elles portent les éléments chimiques qui font partie de l'organisme à d'autres (par exemple les hémoglobines qui portent l'oxygène).

Les protéines se composent de chaîne des acides aminés. Chaque acide aminé a une structure constante. Il y a 20 acides aminés. Deux acides aminés peuvent se joindre, avec un " lien de peptide ", formant une chaîne: un " polypeptide ".

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.

-Exemple d'acide aminé : « Lysine » codé par la lettre « K ».

-Exemple d'une protéine: l'insuline**:**

"F V N Q H L C G S H L V E A L Y L V C G E R G F F Y T P K A"

La synthèse de protéine se produit dans des structures appelées *les ribosomes* situés dans la cellule mais en dehors 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 (Figure 1.12). Comment interprète-on l'information diffusée par ARNm? Ceci est une séquence de "triplets" de nucléotides, ou *de codons.* Chaque codon indique un acide aminé (figure 1**.**13). Mais puisqu'il y a $4^3 = 64$ de codons possibles, mais seulement 20 acides aminés, il y a une certaine redondance dans le code où des triplets différents codent le même acide aminé (voir la table 1.1).

Cette fonction de codage, f: codon à acide aminé est le code génétique elle est universel, pour tous les organismes.

N.B. : les trois codons spéciaux : stop codons; ils ne codent pas un acide aminé; mais ils indiquent la fin d'une région de codage de protéine sur une grande molécule d'ADN.

La traduction : est le processus par lequel une séquence des codons *est traduite* vers une séquence d'acides aminés. Une molécule appelée l'ARN de transfert (ARNt*)* permet le passage des codons aux acides aminés. ARNt contient un triplet appelé *anticodon,* celui-ci possède une extrémité à la quelle un acide aminé spécifique vient s'attacher. ARNt est situé dans le cytoplasme, et porte les acides aminés vers les ribosomes. Les acides aminés rassemblés par les ARNts vont alors collés les uns aux autres pour former une chaîne de peptides appelée polypeptides. Une chaîne de polypeptides peut atteindre une taille de 50 à 30000 acides aminés, la moyenne étant 400 acides aminés.

Après transcription d'ADN et avant la synthèse de protéine, un processus enlève quelques segments de l'ARN (*introns),* laissant seulement les codons significatifs (*exons)* qui seront *ex*primés. Ce processus est appelé «l'*épissage* » [32].

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 [4].

Structure ⟶ ***Fonction***

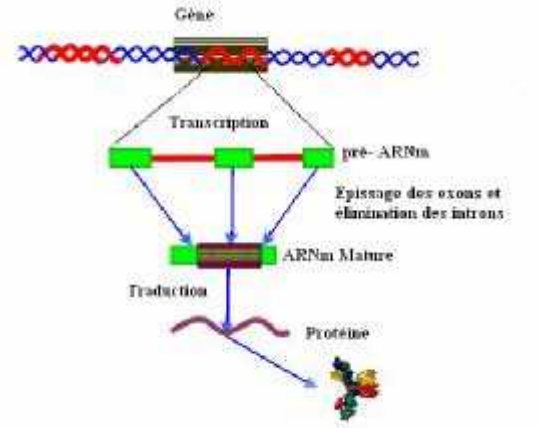


Figure 1.12: Synthèse de protéine

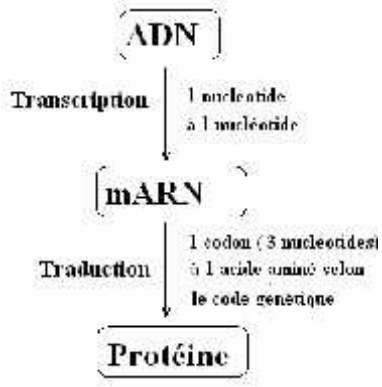


Figure 1.13: Processus du codage

.

Deuxième lettre									
	U		C		A		G		
Première lettre	UUU	Phénil-alanine	UCU	Sérine	UAU	Tyrosine	UGU	Cystéine	U
	UUC		UCC		UAC		UGC		C
	UUA	Leucine	UCA		UAA	Codon stop	UGA	Codon stop	A
	UUG		UCG		UAG		UGG	Tryptophane	G
	CUU	Leucine	CCU	Proline	CAU	Histidine	CGU	Arginine	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Glutamine	CGA		A
	CUG		CCG		CAG		CGG		G
	AUU	Isoleucine	ACU	Thréonine	AAU	Asparagine	AGU	Sérine	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lysine	AGA	Arginine	A
	AUG	Méthionine	ACG		AAG		AGG		G
	GUU	Valine	GCU	Alanine	GAU	Acide aspartique	GGU	Glycine	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Acide glutamique	GGA		A
	GUG		GCG		GAG		GGG		G

Troisième lettre

Table 1.1 : Le code génétique des acides aminés

4.5.Le Gène

Définition: un gène est un fragment de l'information génétique (ADN) correspondant à une protéine. Nous pouvons récapituler ce mécanisme comme « **dogme central** » de biologie moléculaire:

ADN = ARN = protéine = phénotype.

- ✓ la transcription est la propriété de passer de l 'ADN à l'ARN
- ✓ la traduction est le processus de passer de l'ARN à la protéine

N'importe quelle interférence dans ces étapes changerait le phénotype, c.-à-d. la structure et la fonction de l'organisme.

Le génome est un ensemble de tous les gènes d'un organisme donné.

Ce pendant, on sait aujourd'hui qu'à un gène ne correspond pas forcément à une protéine unique. En effet, l'expression peut subir des modifications:

- post-transcriptionnelles: l'ARN messager transcrit à partir d'un gène, peut être recombiné (certaines parties sont coupées et éliminées : les introns, les autres sont "recollées" entre elles : les exons). C'est ce qu'on appelle l'épissage alternatif, qui peut être modulé efonction du cycle cellulaire ou de stimulus extérieurs.
- post-traductionnelles: le repliement 3D d'une protéine peut être modifié, par exemple sous l'action d'une protéine particulière. D'autre part, de nombreuses protéines subissent des modifications chimiques (formation de ponts désulfures, ajouts de groupements sucres pour former des glycoprotéines) après leur synthèse.

Ces variations d'expression sont à l'origine de la complexité de l'expression de l'information génétique. Certes, toute l'information génétique est contenue dans l'ADN, mais deux cellules au même contenu ADN peuvent être extrêmement différentes, en fonction du contenu de leur cytoplasme (différentiation des cellules dans un organisme).

Ceci a permis de mieux ajuster le dogme central de biologie :

- avant 1 gène = 1 ARN = 1 protéine
- maintenant 1 gène = x ARNs = xy protéines

4.5.1. Comment les Génomes Sont Régulés

Chaque cellule dans le corps contient toute l'ADN, et par conséquent la recette pour n'importe quelle protéine. Mais chaque cellule synthétise sa propre protéine. Il y a ici un certain processus de différentiation.

La différentiation de l'ADN commune dans une variété de types de cellules se produit par le fait qu'un gène peut être en état de marche ou arrêt (allumé ou éteint) [29]. Pour déterminer exactement le produit d'une cellule, il faut être capable de répondre aux questions suivantes :

- ✓ Ce qui rend un gène en état de marche ou arrêt
- ✓ Quand est-ce qu'un gène est en état de marche ou arrêt?
- ✓ Où (en quelles cellules) un gène est en marche?
- ✓ Combien de copies du produit gène sont produites?

La réponse à ces questions permettrait aux biologistes de prédire le fonctionnement de n'importe organisme dont on détient le matériel génétique.

4.5.2. Évolution d'un gène

Un gène peut subir des modifications et des opérations dont le résultat est souvent un nouveau gène. C'est cette évolution qui a donné naissance à plusieurs espèces d'organismes [01]. Et qui a participé à l'enrichissement de la nature. On peut citer quelques opérations de

modifications de gènes qui peuvent survenir d'une manière spontanée ou provoquées par des acteurs externes [04]:

- Réplication ou Duplication d'un gène : un gène existant peut se reproduire afin de créer une paire de gènes identiques (division cellulaire).
- Mutation : est définie comme un changement dans la structure d'une séquence d'ADN. C'est la substitution d'un nucléotide par un autre. Ceci peut se produire lors d'une réplication. La mutation peut se manifester à une échelle plus élevée au niveau chromosomique. (voir Figure1.14)
- Insertion : elle est définie comme une insertion d'un nucléotide dans une séquence d'ADN
- Délétion : c'est la disparition d'un nucléotide d'une séquence sans qu'il soit remplacé par un autre.
- Croisement de gènes ou recombinaison : deux gènes peuvent être cassés et puis reliés pour former un nouveau gène hybride composé des segments de l'ADN qui appartenus aux gènes séparés.
- Transfert (intercellulaire) horizontal : un morceau d'ADN peut être transféré à partir du génome d'une cellule à une autre (même d'une espèce à une autre : cas des virus).

Chacune de ces modifications laisse une trace caractéristique dans la séquence d'ADN de l'organisme en affectant son génotype par conséquence son phénotype.

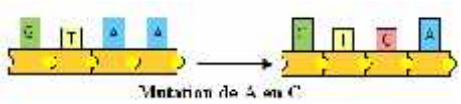


Figure 1.14 : Une mutation d'un nucléotide vers un autre

« Évolution des gènes = mutation, insertions délétions, recombinaison »

Le processus évolutionnaire se produit à différents taux. Si les mutations d'ADN se produisent dans des régions non critiques, elles sont incorporées à la prochaine génération. Si les mutations se produisent dans des régions critiques, elles ont peu de chance d'être propagées dans les générations futures. Cependant, quelques mutations ont des effets positifs, et sont conservées. La conservation des séquences implique la fonctionnalité. Le fait que l'évolution n'a pas modifié une région d'une séquence suppose qu'elle soit fonctionnellement importante pour l'organisme [01]:

- Les régions fonctionnelles des gènes (sites catalytiques, de fixation etc.) sont soumises à la sélection. Elles sont relativement préservées par l'évolution car des mutations trop radicales sont désavantageuses.
- Les régions non fonctionnelles ne subissent aucune sélection et divergent rapidement à mesure que les mutations s'accumulent.

Les nouveaux gènes apparaissent surtout par transmutation des gènes ancestraux : on peut donc déduire la fonction de la plupart des gènes par comparaison avec des gènes « homologues » d'autres espèces dont la fonction est déjà connue.

4.5.3. Homologie et similitude des gènes

Le paradigme central de la bioinformatique est : « ***la déduction par homologie*** ».

Terminologie :

Identité : proportion des paires de bases (résidus) identiques entre deux séquences exprimée généralement en pourcentage.

Similitude : mesure de la ressemblance entre deux séquences. Le degré de similitude est quantifié par un pourcentage de substitutions conservatives des séquences.

Homologie : deux séquences sont homologues si elles ont un ancêtre commun. Il n'y a pas de degré d'homologie. On ne dit pas : très homologues, faiblement homologues. Deux gènes sont homologues ou ils ne le sont pas.

Toutes les opérations modification de gènes citées ou non dans le paragraphe précédent, permettent :

- Spéciation : c'est la séparation d'une espèce en deux, chaque population évolue et forme une nouvelle espèce. Cette modification est le fruit d'une insertion, délétion ou mutation au niveau d'un gène .
- Les nouvelles espèces héritent des mêmes gènes, mais modifiés
- Divergence : leurs gènes accumulent des mutations et génèrent d'autres espèces.

4.5.4. Gènes Orthologues et Paralogues

Deux gènes homologues : signifie qu'ils ont un ancêtre commun.

Deux gènes similaires : implique des protéines similaires puis une fonction similaire.

5. Les Banques de Données Biologiques

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 (Figure. 1.15) 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 [03] 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 [05].

On peut trouver des banques spécialisées pour le stockage des motifs. En outre, des bases de données de la littérature scientifique (telle 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.

Figure 1.15 : Interrogation d'une base de données

5.1.Les Banques de 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), [10] elle est aujourd'hui diffusée par l'EBI (European Bioinformatics Institute, Cambridge, UK).

GenBank : créée en 1982 par la société IntelliGenetics et diffusée maintenant par le NCBI (National Center for Biotechnology Information, Los Alamos, US). [06] elle est soutenue par le NIH (National Institute of Health). Elle possède plus de 50 millions séquences stockées

DDBJ (Dna Data Bank): créée en 1986 et diffusée par le NIG (National Institute ofGenetics,Japon).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.

5.2.Les Banques de Séquences Protéiques

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 ProteinSequences, Munich, Allemagne) et de la banque japonaise JIPID (Japan International Protein Information Database)

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.

5.3.Les Banques de Motif

Prosite : La base de données dédiées aux stockages des motifs protéiques ayant une signification biologique [11] peut être considérée comme un dictionnaire de motifs.

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

6. L'Analyse des Séquences

Les données primaires des projets séquençage sont des séquences d'ADN. Celles-ci sont devenues vraiment exploitables à travers leur annotation. Plusieurs étapes d'analyse avec des outils de la bioinformatique sont nécessaires pour partir d'une séquence d'ADN crue et atteindre des séquences annotées d'une protéine:

- Établir la séquence correcte des fragments contigus d'ADN pour obtenir une séquence continue.
- Trouver les emplacements de déclenchement de transcription et la traduction, trouver des sites de promoteurs, et des ORFs (Open Reading Frame = cadre ouvert de lecture);

- Trouver emplacements d'épissage, introns, exons;
- Traduire la séquence d'ADN en une séquence de protéine
- Comparer la séquence d'ADN à des séquences connues homologues de protéine afin de vérifier les exons… etc.
- Déterminer la structure (surtout la structure tertiaire 3D) puis la fonction de la protéine par comparaison à d'autres séquences semblables.
- Déterminer une origine et/ou une histoire évolutive commune (phylogénie).

Pour un bioinformaticien, une séquence biologique est un MOT ou une chaîne de caractères dont on ne peut manipuler que sa structure primaire présentée généralement dans un format donné. L'analyse des données biologiques consiste en général à chercher un motif dans une séquence, aligner deux ou plusieurs séquences, comparer un motif ou séquence avec les données d'une banque et établir un lien phylogénétique…etc.

7. Conclusion

Dans ce chapitre nous avons présenté le demain d'utilisation de bioinformatique et quelque des notion de base concernant la biologie moléculaire enfin vu Les Banques de Données Biologiques.

CHAPITRE II

Méthodes d’alignement de séquences biologique

1. Introduction

La résolution de différentes sortes de problèmes rencontrés dans notre vie quotidienne a poussé les chercheurs à proposer des méthodes de résolution et à réaliser de grands efforts pour améliorer leurs performances en termes de temps de calcul nécessaire et/ou de la qualité de la solution proposée

Dans ce chapitre, nous allons présentons le problème d'alignement de séquences biologique Pour un jeu de séquences donné. Ensuite nous abordons les méthodes d'alignement de séquences biologique.

2. Alignement Multiple de Séquences

2.1.Présentation

L'alignement multiple des séquences d'ADN ou de protéines est une des techniques les plus utilisées dans l'analyse de séquence. Il est considéré parmi les problèmes les plus difficiles en bioinformatique.

L'alignement multiple de séquences (Multiple Sequence Alignment : MSA) est une tâche cruciale et très importante en biologie moléculaire. MSA offre aux biologistes un moyen pour analyser des séquences d'ADN ou de protéines et de déterminer par la suite leur degré d'homologie ou de divergence. MSA est utilisé dans la construction des arbres phylogénétiques et identifier les motifs dans des familles de protéines, ceci permet de prédire leur aspect structurel et fonctionnel.

La qualité d'une comparaison ou d'une prédiction dépend de la qualité du MSA. Jusque récemment le choix d'une méthode pour la construction des alignements multiples de séquence (MSAs) a été limité à une poignée de packages mais une augmentation récente des données génomique a poussée l'élaboration de plusieurs nouvelles méthodes, plus précises et plus rapides que les anciennes. Dans la pratique, ce large choix a également rendu difficile le choix objectif de la méthode appropriée pour un problème spécifique.

Pendant la dernière décennie, plus de 50 méthodes ont été décrites dans ce domaine et 20 uniquement pendant l'année 2005 [34]. Ce nombre risque d'augmenter car aucune parmi elles n'est totalement efficace pour tout type de séquences.

Pour étudier l'évolution de gène à travers un éventail d'organismes, les biologistes ont besoin des outils précis pour l'alignement multiple de séquences des familles de protéines. L'obtention des alignements précis, cependant, est un problème informatique difficile en raison non seulement du coût informatique élevé mais également du manque de fonctions objectives appropriées pour la qualité de mesure d'alignement. Il a été démontré que MSA est un problème NP-Complet [33]. Donc la résolution d'un MSA par une méthode exacte paraît

une mission difficile voire impossible. Les méthodes proposées dans la littérature sont en général des heuristiques qui tentent d'approcher un alignement optimal sans l'atteindre réellement ceci est dû à la complexité des données biologiques.

Dans ce chapitre, nous allons commencer par exposer les principales fonctions objectif utilisées puis les méthodes les plus récentes conçues pour résoudre le problème de MSA selon les approches utilisées.

2.2.Définition formelle d'un Alignement Multiple

Un alignement multiple de séquences est en réalité un agencement de plusieurs séquences biologiques dans le but de mettre en valeur leur similitude et convergence.

Un alignement multiple dépend du nombre de séquences ainsi que de leur longueur. Un MSA est souvent facile à réaliser lorsque les séquences sont issues de la même famille dans le cas contraire, séquences divergentes, MSA devient très délicat car il est difficile de repérer les zones La problème vu au chapitre peut être généralisé pour un ensemble de k séquences, avec k > 2. On parle alors d'alignement multiple de séquences. Les définitions ainsi que les propriétés vont également pouvoir être généralisées.

Définition 2.1 : Soit S un ensemble de k > 2 séquences, et soit A un alignement multiple de S. Le problème d'alignement multiple de séquences consiste à déterminer si pour une fonction d'évaluation, A est le meilleur alignement multiple possible des séquences de S.

Définition 2.2 : Soit Σ un alphabet sans le caractère '-' et Σ' = Σ ¨ {'-'}, en plus, soient *S1,… , Sk* les K séquences sur Σ avec des longueurs n1,…… , nk. Soit A l'alignement multiple de *S1,… , Sk. A* est une matrice de dimension K ·L avec les propriétés suivantes [27]:

- Max {n1,…,nk } ≤ L ≤ $\sum_{i=1}^{k}$ ni
- A[i][j] ∈Σ' ∀ 1 ≤ i ≤ K; 1 ≤ j ≤ L.
- La i^{ème} ligne Ai sans gap est égale à Si.
- Il n'a y a pas de colonnes ne contenant que de gaps.

2.3.Les Utilisation en bionformatique

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.

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.

2.4.Evaluation de MSA et Fonctions Objectif

En général, l'alignement optimal est celui qui optimise une fonction objectif (FO). Une fonction objectif ou méthode de score est une expression mathématique qui essaye d'attribuer une évaluation quantitative à la signification biologique et évolutionnaire d'un alignement.

Ainsi, ces méthodes tentent de trouver le MSA optimal qui maximise ou minimise une FO. Le choix d'une FO peut s'avérer une tâche très délicate car le problème est purement biologique. Comment s'assurer mathématiquement qu'un alignement est correct biologiquement ? D'où l'apparition d'un nombre non négligeable de F.Os qui tentent toutes de définir un alignement optimal mathématiquement, mais malheureusement l'optimum mathématique coïncide rarement avec l'optimum biologique [19] mais les fonctions objectif nous permettent de s'approcher de celui-ci.

Toutes les méthodes de score essayent de donner une évaluation quantitative à la signification biologique et évolutionnaire d'un alignement. Cependant, en raison de la nature complexe des données biologiques, toutes les méthodes de score ont leurs limitations. Il n'y a aucune norme universelle pour mesurer la qualité d'un alignement multiple de séquences.

Définition2.3 : En général, une fonction objectif est une expression mathématique qui permet d'évaluer la qualité d'un résultat d'un traitement.

Dans notre cas, elle va servir à l'évaluation des alignements multiples obtenus et de décider lequel est meilleur.

2.4.1. La Somme des Paires (Sum of Pairs : SP)

C'est la fonction la plus répandue, elle est simple à réaliser.

Elle consiste à sommer les scores des paires de séquences alignées dans un alignement multiple Ai avec Ai (ceci par référence aux définitions précédentes).

Soit Ai un alignement de K séquences $\{S1, \ldots, Sk\}$;

$$SP(Ai) = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} Sc(SiSj).$$

Avec Sc(*SiSj*) est le score de l'alignement de la paire des séquences Si et Sj. Ce score peut être calculé par une mesure de distance ou de similitude.

Généralement, on utilise une fonction d'identité qu'il faut bien sûr la maximiser ou une fonction de distance que l'on doit minimiser.

2.4.2. Weighted Sum of Pairs (WSP)

C'est une amélioration de SP, introduite par [02]. Elle consiste à attribuer des poids aux différentes paires de séquences à aligner. Ces poids peuvent être obtenus en dressant un arbre phylogénétique reliant ces séquences selon leurs similitudes et distances. Deux méthodes sont souvent utilisées pour la construction d'un tel arbre et qui sont N.J [22] et UPGMA [23] présentées dans le chapitre précédent.

Le choix d'une ou de l'autre méthode dépend de la nature des séquences à aligner ;

si celles-ci ont plus ou moins de similitudes entre elles.

La formule générale de cette fonction est :

$$WSP(A) = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \boldsymbol{Wij} * \boldsymbol{sc(SiSj)}.$$

2.4.3. La Fonction Consensus

D'après Martin Tompa [31], il est parfois préférable de passer par une séquence consensus ***C*** pour évaluer la qualité d'un alignement multiple de plusieurs séquences. L'idée consiste donc à trouver une séquence C : $c_1c_2c_3\ldots c_L$ où L est la longueur de l'alignement, de telle sorte que chaque caractère c_i de ***C*** minimise le score de celle c_i.

Définition2.4:Ayant un alignement de N séquences $S_1S_2S_{3\ldots}S_N$ le caractère consensus c d'une colonne i,i=1,..L, est celui qui minimise la somme des distances entre lui et les autres caractères de cette colonne $d(i) = \sum_{j=1}^{N} d(Sj'[i], ci)$.

avec S' est S alignée. La séquence consensus étant C : $c_1c_2c_3\ldots c_L$ les erreurs d'alignements sont définis par $\sum_{i=1}^{L} d(i)$

2.4.4. La Fonction Profil

Cette fonction a pour objectif de calculer le profil d'un alignement A [30]. Ce profil est une représentation numérique d'un MSA qui représente les caractéristiques communes d'une famille de protéines. La fonction Profil est utilisée pour déterminer le degré d'appartenance d'une protéine à une famille. On peut signaler qu'il est utile dans l'alignement des séquences pas trop divergentes. Il permet de déterminer des régions conservées dans une séquence ou plusieurs. C'est la somme des fréquences d'apparition de chaque résidu dans chaque colonne de l'alignement.

2.4.5. La Mesure d'Entropie

La mesure d'entropie en MSA est la somme d'entropie des colonnes [20].

L'entropie est en générale une mesure de variation des informations utilisée souvent dans la théorie de l'information introduite par **Shannon**. Pour chaque colonne, l'entropie est calculée par la formule suivante : $Entropie\ (A[:][i]) = -\sum_a Cia * \log(Pia)$

2.4.6. La Fonction Coffee

Coffee (Consistency-based Objective Function For alignmEnt Evaluation) [18].Elle fournit un score global de l'alignement, appliquée pour l'évaluation les alignements produits par la méthode SAGA [17].

Cette fonction a la particularité d'avoir utilisé un nouveau concept : 'Consistency' en anglais dont la signification dans ce contexte est 'Consistance'. Ce concept fut introduit la première fois par .

3. Les Approches de résolution de MSA

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 [08].

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.

Les méthodes qui suivent l'approche progressive, sont reconnues d'être très rapides [28] 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.

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 [35].

3.1.L'Approche Exacte

L'approche exacte n'est autre qu'une généralisation des méthodes de programmation dynamique de [24]

La méthode de programmation dynamique utilisée pour aligner deux séquences, a été appliquée à l'alignement de plusieurs séquences (N dimensions) tels que MSA DCA [25].

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

- ✓ Dans la pratique, un alignement devient délicat pour un nombre de séquence $N > 3$, et même impossible pour $N = 10$

- ✓ Pour N séquences de longueur L, l'alignement optimal (au sens mathématique) nécessite :
  - o Un temps de calcul proportionnel à $2^n L^n$
  - o 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 = $2^{10} * 100^{10} * 10^{-9}$ 10^{14} s ($>9^{10}$années)

Espace mémoire = 10 11 6 GB.

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

3.2.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.

Les étapes d'un alignement itératif :

- ✓ Repérer les deux séquences avec la plus forte similarité et les aligner avec une méthode de programmation dynamique.
- ✓ Trouver la séquence qui est la plus proche du profil obtenu avec les 2 séquences précédentes et l'aligner avec les deux autres par une méthode d'alignement profil-séquence.
  - o Répéter ceci jusqu'à ce que toutes les N séquences soient inclues dans l'alignement multiple
- ✓ Enlever la séquence S1 et la réaligner avec le profil obtenu avec les séquences de S2…Sn
  - o Répéter ceci pour toutes les autres séquences de S2 à Sn.
- ✓ Répéter l'étape précédente un certain nombre de fois ou arrêter le processus à convergence du score de l'alignement.

3.3.L'Approche Progressive

L'alignement progressif [Taylor, 87] 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 [07], ClustalW [31], et T-Coffee [18] …etc. Un alignement multiple progressif suit les étapes suivantes [09]:

Un alignement multiple progressif suit les étapes suivantes [09]:

- 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 :

- 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.

4. Les Méthodes d'Alignement Multiple Exactes

4.1.La méthode MSA

C'est une tentative de rendre les algorithmes de la programmation dynamique conçus pour aligner deux séquences, opérationnels pour un alignement multiple. Sachant que la complexité temporelle et spatiale augmente proportionnellement avec le nombre et la longueur des séquences à aligner. Même la matrice de score devient elle aussi multidimensionnelle.

Le programme de **MSA** emploie un algorithme intelligent pour réduire le volume de la matrice de la programmation dynamique multidimensionnelle. L'algorithme de Carrillo et de Lipman était mis en application dans MSA.

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

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

En plaçant la limite inférieure et la limite supérieure, seulement un espace restreint doit être exploré dans la table de score multidimensionnelle [13]. Toutes ces considérations ont participé à la réduction du temps de calcul d'une manière significative.

Ce type de méthodes représente de gros problèmes : Le temps de calcul et l'espace mémoire. Dans la pratique, un alignement devient délicat pour un nombre de 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 2n Ln.
-Un espace mémoire proportionnel à Ln.
Le problème de l'alignement multiple exacte a été démontré être un problème NP-complet. D'où Le recours aux méthodes approchées ou heuristiques.

4.2.la Méthode de DCA

DCA (Divide and Conquer Algorithm) [25], c'est une heuristique basée sur l'idée « diviser puis conquérir ». Le principe consiste à découper les séquences à aligner en sous-ensembles de segments. Ces segments doivent avoir une taille assez petite pour faciliter leur traitement par MSA. Les sous alignements produits sont alors concaténés pour former un seul alignement multiple final.
Comme étant une méthode exacte, elle hérite des mêmes inconvénients des méthodes de ce type : la complexité temporelle et spatiale.

5. Les Méthodes d'Alignement Multiple itératifs

5.1.La Méthode SAGA

C'est un algorithme génétique itératif [19] qui démarre par une population d'alignement, puis raffine les solutions par des opérateurs spécifiques tels que la mutation jusqu'à l'obtention d'une solution plus ou moins optimale. C'est une heuristique qui se rapproche de la solution optimale mais aucune certitude qu'elle le soit réellement.

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

Chaque génération est évaluée par la fonction objectif (*WSP*) pour déterminer quels sont les alignements les plus acceptables et aptes à passer dans la génération suivante. Ceci est appelé le phénomène de la sélection biologique *« seuls les meilleurs survivent »*.
G0, Gn et Gn+1 sont respectivement la population initiale, courante et la population de la génération future. L'algorithme commence par la génération des individus de la population

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

SAGA a la particularité de pouvoir optimiser n'importe quelle fonction objectif. Plus tard [17] ont utilisé SAGA pour valider une nouvelle fonction objectif : Coffee.Les résultats sont considérés nettement meilleurs que ceux fournis par la première approche.

5.2.La Méthode DIALIGN

DIALIGN est une méthode pour l'alignement multiple développée par [16]. L'algorithme de DIALIGN est basé sur les alignements par paires de séquence (alignement deux à deux) et multiple en comparant des segments entiers de séquences au lieu d'une traditionnelle comparaison de chaque résidu.

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

Une fois une diagonale est considérée dans un alignement, elle est fixe et ne peut pas être enlevée à une étape postérieure de l'algorithme. Une diagonale n'est pas choisie selon son poids, mais plutôt selon si le motif décrit par cette diagonale, apparaît dans plus de deux séquences, alors il est préféré aux motifs qui apparaissent dans seulement deux séquences. Cette approche est particulièrement efficace et convenable pour la détection d'une homologie locale. Sa consommation en termes de durée de calcul et en espace mémoire est considérée raisonnable [14]. Dialign-t [26] est une version plus récente de Dialign-2, locale et progressive.

6. Les Méthodes d'Alignement Multiple Progressives

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.

6.1.La Méthodes ClustalW

ClustalW [31] est un programme qui met en action les principes de l'alignement progressif tout en essayant d'échapper au piège des erreurs qui peuvent se produire au début de l'alignement et nuire à sa qualité dans la fin.

Dans ClustalW, les auteurs essayent donc de respecter la démarche progressive mais en apportant des modifications et des nouvelles considérations.

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

Dans la deuxième étape, ClustalW utilise la méthode N.J [22] pour construire un arbre guide et calculer les poids des séquences.

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

Dans ClustalW, il y a une grande étude et des nouvelles propositions sur la manière de faire changer les valeurs affectées à un gap selon sa position dans une séquence ou dans un alignement de séquences. Une particularité de ClustalW est qu'il possède une interface graphique conviviale contrairement aux autres méthodes

6.2.La Méthode MUSCLE

La méthode MUSCLE [Edgar, 04] 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.

La méthode MUSCLE peut être décrite en trois étapes essentielles :

L'étape 1 : Le but de la première étape est de produire rapidement un alignement multiple avec plus d'exactitude possible. Ceci est basé sur la détermination d'une matrice D1 de distances à partir de la distance de k-mers entre toutes les paires de séquences.

La matrice obtenue est alors clustérisée par UPGMA, pour produire un arbre binaire TREE1. Un alignement progressif MSA1 est construit alors en suivant l'ordre dicté par l'arbre.

Etape 2 : La source d'erreur principale à l'étape progressive est la mesure approximative de distances k-mer, qui a comme conséquence un arbre sous optimal. MUSCLE re-estime donc

l'arbre en utilisant la distance de Kimura, qui est plus précise mais exige l'utilisation un alignement dans ce cas c'est MSA1 donnant ma matrice D2. D2 va subir le même procédé de clustérisation afin de produire un arbre binaire TRRE2 et progressivement construire l'alignement MSA2

Etape 3 : C'est une étape d'amélioration. TREE2 est divisé en deux sous arbres en supprimant la branche qui les relient. Celle-ci est choisie en parcourant l'arbre à partir de la racine. Le profil de l'alignement multiple dans chaque sous arbre est alors calculé. Un nouvel alignement multiple est produit en réalignant les deux profils. Si le score de PS est amélioré, le nouvel alignement est gardé, autrement il est rejeté et l'étape 3 est alors répétée jusqu' à la convergence ou jusqu'à ce que une limite définie soit atteinte.

Considérée la plus rapide et plus exacte, la méthode MUSCLE est la plus répandue actuellement avec ClustalW.

7. 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.

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. [15] 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.

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. Plusieurs bases d'alignements de référence ont été élaborées comme BaliBase [31], et OxBench [29] Prefab[08], SABmark [32].

- **La Base de référence ' BaliBase'**

BAliBASE [31] se compose de 142 alignements de référence, contenant plus de 1000 séquences avec 200.000 résidus. Les alignements sont divisés en cinq catégories hiérarchiques de référence. Chacune des catégories peut être encore subdivisée en plus petits groupes, selon la longueur de séquence et les pourcentages de similitude :

La référence 1 : contient des alignements de (moins de 6) séquences équidistantes, dont le pourcentage d'identité entre deux séquences est dans une marge indiquée. Toutes les séquences sont de longueur semblable, pas de grandes insertions ou prolongements de gaps.

La référence 2 : aligne jusqu'à trois séquences orphelines (moins de 25% d'identité) de la référence 1 avec une famille d'au moins de 15 séquences très proches.

La référence 3 : se compose de familles de séquences équidistantes divergentes (un pourcentage d'identité <25%).

La référence 4 : est divisée en deux sous-catégories contenant des alignements de jusqu'à 20 séquences comprenant des prolongements de N/C-terminal (jusqu'à 400 résidus),

La référence 5 : contient des séquences de longues insertions internes (jusqu'à 100 résidus).Dans BAliBASE des blocs « *noyau»* (cores) sont annotés pour les alignements qui incluent des régions qui doivent être correctement alignées. Les blocs excluent des régions où il y a une possibilité d'ambiguïté. Ceci peut être un facteur important affectant la signification des comparaisons statistiques des programmes d'alignement.

Pour évaluer la qualité d'un alignement, BaliBase offre un programme en C, Bali-Score quipermet d'évaluer le résultat d'un programme en comparant l'alignement fourni avec le jeu de test correspondant. En résultat, Bali-Score édite deux valeurs qui déterminent la qualité del'alignement qui sont la valeur SPS (Sum of Pairs Score) qui indique le nombre de paires de résidus correctement alignées et CS (Columns Score) qui test l'habilité du programmes à aligner toute la séquence et qui indique le nombre de colonnes correctement alignées.

Actuellement, BaliBase est la base de référence la plus utilisée pour les tests protéiques. D'un autre côté, il existe une base dédiée aux alignements des séquences nucléiques et qui est *BraliBase* [11].

8. Conclusion

dans ce chapitre nous avons présenté la problème de MSA(Multiple Sequence Alignement) et les Fonction d'évaluations, en suite on a abordé les déférant approches de résolution de problème MSA ainsi que les méthodes de chaque approche enfin on a cité les déférant référence utilisé sur le BALIBASE comme un référence d'évaluation.

CHAPITRE III

Optimisation par Métaheuristiques à population

1. Introduction

La résolution de problèmes d'optimisation constitue un axe de recherche qui a sollicité l'attention de plusieurs équipes de recherche, vu l'importance capitale qu'elle revêt dans notre vie quotidienne. Cette branche de recherche fait appel à des astuces mathématiques et informatiques. Elle est intrinsèquement liée à la recherche opérationnelle, l'algorithmique et la théorie de la complexité. La résolution d'un problème d'optimisation consiste à rechercher solution d'une qualité suffisante parmi un ensemble de solutions au regard d'un (des) critère(s) donné (s) et des objectifs à satisfaire. Elle consiste à maximiser ou à minimiser une ou un ensemble de fonctions fitness en respectant des contraintes liées au problème traité. La résolution de n'importe quel problème d'optimisation s'effectue par un procédé algorithmique de complexité spatiale et temporelle qui selon leur degré, elles permettent de classer les problèmes d'optimisation en différentes classes (P, NP, NP-Complet et NP-Difficile).

Les méthodes de résolution de problèmes d'optimisation sont nombreuses. Elles sont souvent classées en deux grandes classes: La classe des méthodes exactes et la classe des méthodes approchées. Les méthodes exactes garantissent l'optimalité de la solution, mais elles sont souvent exigeantes en termes de temps de calcul nécessaire. Notamment, pour la résolution des problèmes de grande taille. Pour pallier à cette lacune, les méthodes approchées sont considérées actuellement comme un bon choix de résolution. L'investigation dans le domaine des méthodes approchées a donné naissance à une autre catégorie de méthodes appelées « Métaheuristiques ». Les métaheuristiques sont des méthodes générales et applicables sur une grande gamme de problèmes d'optimisation. Elles sont souvent inspirées des systèmes naturels dans différents domaines: physique, biologie, éthologie…etc.

Ce chapitre est réservé à la description détaillée de deux métaheuristiques à population : La méthode d'optimisation par essaim de particules et les algorithmes génétiques.

2. Optimisation par Algorithme PSO

2.1. Présentation

L'idée de s'inspirer des systèmes naturels pour proposer des méthodes de résolution des problèmes d'optimisation a donné naissance à une sous classe des métaheuristiques, ce sont des métaheuristiques basées sur l'intelligence par essaim (en anglais: Swarm Intelligence). La métaheuristique d'optimisation par essaim de particules (PSO: Particle Swarm Optimization) est la métaheuristique principale dans la classe des métaheuristiques basées sur l'intelligence par essaim. C'est une des métaheuristiques à base de population de solutions inspirées par une analogie avec l'éthologie.

L'optimisation par essaim de particules est une jeune métaheuristique, elle imite le comportement social des animaux évoluant en groupe comme les oiseaux, les abeilles et les poissons. Le PSO a été proposé en 1995 par Kennedy et Eberhart pour la résolution des problèmes d'optimisation continus. La simplicité et la performance de cette méthode ont suscité l'intérêt de plusieurs communautés de chercheurs qui ont mené des études d'optimisation et d'application de cette métaheuristique pour la résolution de plusieurs problèmes d'optimisation continus et/ou discrets. Par conséquent, plusieurs alternatives de l'algorithme originel du PSO ont été proposées dans la littérature afin d'améliorer sa performance pour la résolution de différents problèmes.

2.2. L'origine de l'idée de l'optimisation par essaim de particules

L'idée de l'optimisation par essaim de particules trouve ses racines dans les années 80. Précisément en 1983, lorsque Reeves [38] a essayé de résoudre le problème de rendu des images afin de simuler les phénomènes naturels en utilisant l'outil Informatique pour créer des scènes animées. Dans le cadre de son travail, Reeves a implémenté un système de particules qui œuvrent ensemble pour simuler un objet flou (nuage, explosion…). Le modèle proposé par Reeves considère que chaque particule est caractérisée par une position dans l'espace de recherche et une vitesse de déplacement. En effet, les particules de l'essaim se déplacent en fonction de leurs positions courantes et leurs vitesses qui seront adaptées au cours de la recherche.

D'autre part, Craig Reynolds [39] a été intrigué par l'organisation et l'esthétique du comportement social des oiseaux. Il a tenté d'améliorer l'idée de Reeves, en rendant le comportement du groupe des particules plus dynamique et plus organisé. Reynolds a ajouté la notion d'orientation et la notion de communication inter -particules: chaque particule doit rester proche des autres particules de l'essaim comme elle (i.e. la particule) doit éviter d'entrer en collision avec ses congénères. C'est la raison pour laquelle, chaque particule doit avoir conscience de la position des autres particules du groupe. Suite à sa recherche, Reynolds a découvert que l'implémentation d'un modèle simulant le comportement de particules tel qu'il est en réalité n'est pas faisable. En fait, il a découvert que son modèle engendre une exécution très complexe surtout avec une population de grande taille. Afin de pallier à ce problème, Reynolds a proposé l'utilisation de la notion du voisinage. De leurs parts, Heppner et Grenander [37] ont apprécié la manière de volées d'oiseaux. Ils ont établi une recherche sur les différentes règles permettant à un ensemble d'agents de se communiquer d'une manière très simple et de produire un comportement social imprévisible, bien organisé et intelligent. En 1995, Kennedy et Eberhart se sont basés sur les idées et les études de

Reeves, Reynolds et surtout des résultats de Heppner et Grenander afin de comprendre la stratégie de recherche de nourriture et d'affrontement des prédateurs que l'on retrouve chez les groupes d'animaux tels que les bancs de poissons, les volées d'oiseaux ou les essaims d'insectes. Le fruit des recherches de Kennedy et Eberhart était la proposition de la métaheuristique d'optimisation par essaim de particules dont le principe sera décrit dans ce qui suit.

2.3. L'optimisation par essaim de particules (PSO)

L'optimisation par essaim de particules (le PSO en anglais: Particle Swarm Optimazation) est une métaheuristique à base de population de solution. Elle a été proposée en 1995 par Kennedy et Eberhart [40]. L'algorithme PSO est inspiré du comportement social d'animaux évoluant en essaim, tels que les poissons qui se déplacent en bancs ou les oiseaux migrateurs. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une intelligence limitée et une connaissance seulement locale de sa situation dans l'essaim. L'intelligence globale de l'essaim est donc la conséquence directe des interactions locales entre les différentes particules de l'essaim. La performance du système entier est supérieure de la somme des performances de ses parties. Kennedy et Eberhart se sont inspirés de ces comportements sociaux pour créer l'algorithme PSO. Contrairement aux autres algorithmes évolutionnaires tel que l'algorithme génétique où la recherche de la solution optimale évolue par compétition entre les individus en utilisant des opérateurs de croisements et de mutations, le PSO utilise plutôt la coopération entre les individus.

La méthode d'optimisation par essaim particulaire met en jeu un ensemble d'agents pour la résolution d'un problème donné. Cet ensemble est appelé essaim. L'essaim est composé d'un ensemble de membres, ces derniers sont appelés particules. Les particules de l'essaim représentent des solutions potentielles au problème traité. L'essaim de particules survole l'espace de recherche, en quête de l'optimum global. Le déplacement de chaque particule est influencé par les trois composantes suivantes [36] (Figure 3.1):

- Une composante physique: la particule tend à suivre sa direction de déplacement courante;
- Une composante cognitive: la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée;
- Une composante sociale: la particule tend à se diriger vers le meilleur site déjà atteint par ses voisines.

Chaque particule i de l'essaim est définie par sa position $x_{id}= (x_{i1}, x_{i2},\ldots, x_{id},x_{iD})$ et sa vitesse de déplacement $v_{id}= (v_{i1}, v_{i2},\ldots, v_{id},\ldots, v_{iD})$ dans un espace de recherche de dimension D.

Cette particule garde en mémoire la meilleure position par laquelle elle est déjà passée et la meilleure position atteinte par toutes les particules de l'essaim, notées respectivement:
p_{bestiD} =(p_{besti1}, p_{besti2} ,…, p_{bestid},…, p_{bestiD}) et g_{best} = =(g_{besti1}, g_{besti2} ,…, g_{bestid},…, g_{bestiD})
Le processus de recherche est basé sur deux règles :
- Chaque particule est dotée d'une mémoire qui lui permet de mémoriser la meilleure position par laquelle elle est déjà passée et elle a tendance à retourner vers cette position.
-Chaque particule est informée de la meilleure position connue au sein de son voisinage et elle a toujours tendance de se déplacer vers cette position.

La particule i va se déplacer entre les itérations t et t+1, en fonction de sa vitesse et des deux meilleures positions qu'elle connaît (la sienne et celle de l'essaim) suivant les deux équations suivantes [40] :

$$v_{id}(t) = v_{id}(t-1) + c_1 r_1 (p_{bestid}(t-1) - x_{id}(t-1)) + c_2 r_2 (g_{bestd}(t-1) - x_{id}(t-1)) \quad (3.1)$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \quad (3.2)$$

- Avec : xid(t), xid(t-1) : la position de la particule i dans la dimension d aux temps respectivement.
- vid (t), v (t-1) : la vitesse de la particule i dans la dimension d aux temps t et t-1 respectivement.
- *pbestidid* (t-1) : la meilleure position obtenue par la particule i dans la dimension d au temps t1.
- *gbestd* (t-1) : la meilleure position obtenue par l'essaim dans la dimension d au temps t-1.
- c1, c2 : deux constantes qui représentent les coefficients d'accélération, elles peuvent être non constantes dans certains cas [Ratnaweera et al, 2004] et [Kuo et al, 2007].
- r1, r2 : nombres aléatoires tirés de l'intervalle [0,1].
- vid (t-1), $c_1 r_1 (p_{bestid}(t-1) - x_{id}(t-1))$, $c_2 r_2 (g_{bestd}(t-1) - x_{id}(t-1))$: représentent Respectivement, les trois composantes citées au-dessus.

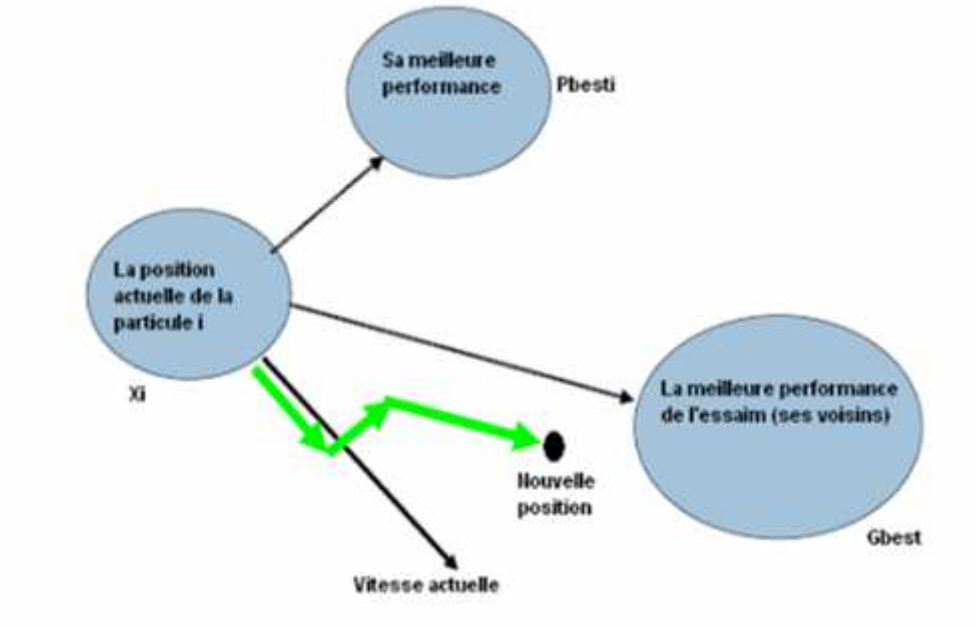


Figure 3.1 : Déplacement d'une particule

Afin d'estimer la qualité de la particule i, il est indispensable de calculer sa fonction «objectif» (aussi appelée fitness). La valeur de la fonction «objectif» de la particule x est notée $f(x_{id})$. Cette dernière est calculée en utilisant une fonction spéciale au problème traité Afin de mettre à jour les valeurs de x_{id} id, p_{bestid} et g_{bestid} leurs fitness sont calculées à chaque itération de l'algorithme. xid bestd est mise à jour selon l'équation (3.2). pbestid et gsont mises à jour si les conditions (C1) et (C2) présentées ci-dessous sont vérifiées respectivement:

$f(x_{id})$ est meilleur que $f(p_{bestid})$ (C1)

$f(p_{bestid})$ est meilleur que $f(g_{bestid})$ (C2)

L'algorithme PSO commence par initialiser la taille de l'essaim ainsi que les différents paramètres, affecter à chaque particule une position et une vitesse initiales et initialiser les p_{bestid} . Ensuite, calculer les fitness des particules afin de pouvoir calculer la meilleure position trouvée par l'essaim g_{bestid} . A chaque itération du processus de la recherche, les particules se déplacent en fonction des équations (3.1) et (3.2). Leurs fitness sont calculées, les p_{bestd} $best_{id}$ et la g_{best} sont mises à jour. Le processus est répété jusqu'à la satisfaction du critère d'arrêt.

L'algorithme 3.1 représente un pseudo code de l'algorithme PSO.

Table 3.1 : pseudo code de l'algorithme PSO

Début

Initialiser les paramètres et la taille S de l'essaim;

Initialiser les vitesses et les positions aléatoires des particules

dans chaque dimension de l'espace de recherche;

Pour chaque particule p **faire**

Calculer f (x_{id}) de chaque particule;

Calculer g_{bestid}; // la meilleure p

Tant que (la condition d'arrêt n'est pas vérifiée) **faire**

Pour (i allant de 1 à S) **faire**

Calculer la nouvelle vitesse à l'aide de l'équation (3.1) ;

Trouver la nouvelle position à l'aide de l'équation (3.2) ;

Calculer f (x_{id}) de chaque particule;

Si (f (x_{id}) est meilleur que f ($\boldsymbol{p}_{bestid}$)) **alors** $\boldsymbol{P}_{bestid} - x_{id}$;

Si (f($\boldsymbol{p}_{bestid}$)) est meilleur que f ($\boldsymbol{g}_{bestid}$)) **alors** $\boldsymbol{g}_{bestd} = \boldsymbol{p}_{bestid}$;

Fin pour

Fin tant que

Afficher la meilleure solution trouvée $\boldsymbol{g}_{bestid}$

Fin

Dans le PSO, chaque particule est influencée à la fois par ses propres informations acquises par les expériences passées et par celles de l'essaim. Malgré les différences remarquables entre le principe directeur de l'algorithme PSO et celui des algorithmes évolutionnaires en général, l'optimisation par essaim de particules fait tout de même partie de cette classe. Wilson [41], démontra que le partage d'informations entre les individus d'un même groupe était, dans certains cas beaucoup plus avantageux pour la survie de l'espèce que la compétition entre les individus. La coopération est ainsi un mécanisme évolutif au même titre que la sélection, le croisement et la mutation. De plus, les mécanismes de sélection, de croisement et de mutation sont bien présents mais sous une forme moins implicite [42].

2.4. Application du méthode PSO au problème MSA

Le problème d'alignement Multiple de séquences avec des tailles et des longueurs considérables est considéré comme un problème d'optimisation a pour but de maximiser la fonction de score. Et comme les méthodes exactes sont reconnues par leur incapacité de résoudre ce type de problèmes dans un temps raisonnable, nous avons opté par l'algorithme d'optimisation par essaim de particules pour trouver une solution approchée au problème posé. Dans cette section on va détailler les différents paramètres de l'algorithme PSO en prenant en compte la spécificité du problème MSA. La version de l'algorithme PSO notée par la suite PSOAlign que nous avons adaptée à la résolution du problème MSA est décrite dans la table 3.1 ci-dessous :

Table 3.2 : Pseudo-code de PSOAlign

Début

1. Générer l'ensemble de particules initiales.
2. Déterminer le leader de particule *gbest*
3. **Répéter jusqu'à** la terminaison des critères est fait.
   3.1 Calcule la distance entre *gbest* et chaque particule.
   3.2 Mouvement de chaque particule vers *gbest*.
   3.3 Déterminer le chef de particule *gbest*.
4. Afficher *gbest* comme solution finale

Fin

L'idée principale de cette algorithme est que l'ensemble des particules qui représentent des solutions admissible déplacent progressivement dans l'espace de recherche pour générer une meilleure solution au problème, ce processus est itérativement répéter jusqu'à aucune amélioration n'est possible.

2.4.1 Représentation de données

Généralement l'essaim est composé d'un ensemble des particules, parmi cet ensemble une particule est désigné en tant que chef (leader) de l'essaim dont sont score est maximal (*gbest*). De plus, chaque particule préserve dans sa mémoire sa meilleure position précédente (*pbest*)
Dans notre algorithme PSOAlign, une particule est correspond à un alignement composé d'un ensemble des séquences de même taille. Ceci implique l'ajout aléatoire d'un ensemble d'espaces appelés « gap » ou « brèche » à condition que le nombre de brèches dans chaque

séquence ne pas dépasser 20% de la taille de séquence la plus longue. Un exemple d'illustration de ce processus est donné dans la figure 3.2 ci-dessous :

W	G	K	V	N	V	D
W	D	K	V	N		
S	K	V	G	G	N	

W	G	K	V	-	-	N	V	D
W	D	K	V	-	-	N	-	-
S	-	K	V	G	G	N	-	-

Figure 3.2 : Exemple de représentation d'une particule.

2.4.2 Population initiale

La taille de l'essaim (c-à-d le nombre de particules) est déterminé par l'utilisateur. Pour générer la population initiale, il consiste à répéter le processus d'insertion des brèches d'une manière aléatoire dans chacune des séquences de l'alignement jusqu'à la taille de l'essaim est atteinte.

2.4.3 Fonction de score

Le score global d'alignement est basé sur le score d'alignement de chaque paire de séquences. Ainsi, chaque séquence doit être alignée avec chaque autre séquence. En générale, le score assigné à chaque particule (alignement) est la somme des scores d'alignement de chaque paire de séquences. Le score de chaque paire de séquences et la somme assigné pour la comparaison entre chaque paire de caractères, qui est donnée par la matrice de BLOSUM62 présentée dans la figure 3.4. Cette matrice contient tous les valeurs associées aux identités/substitutions possibles des différents caractères. La fonction de score retenue dans notre travail est donnée par :

$$\boldsymbol{SP(Ai) = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} Sc(Si,Sj) + Pinalité(gap)}.$$

Où : $Sc(A_i, A_j) = BLOSUM62(A_i, A_j)$

La pénalité est calculée en fonction des brèches d'ouverture (*GOP : Gap Open penalty*) et celles d'extension *(GEP : Gap Extension Penalty*). Notons ici que les valeurs de ces deux brèches adéquates à la matrice BLOSUM62 sont respectivement *Gop*=-10 et *GEP*=-1.

Un exemple de calcul de score de deux séquences est illustré dans la figure 3.3 ci-dessous :

Figure 3.3 : exemple de calcul de score.

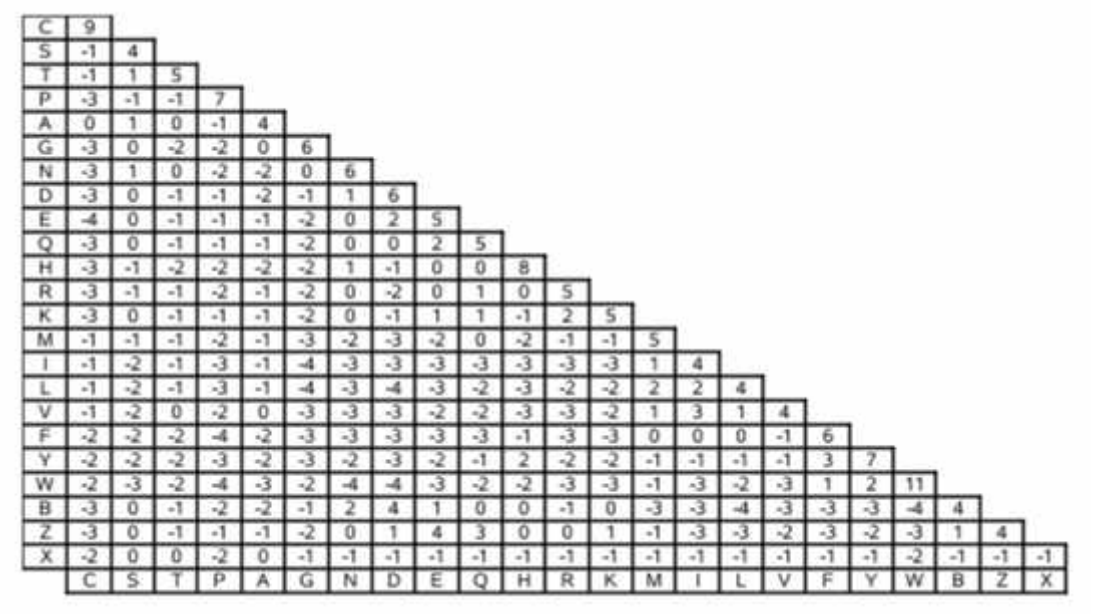


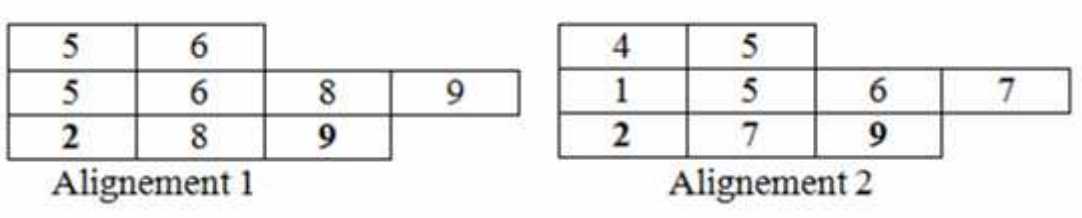
Figure 3.4 : Matrice de BLOSUM62.

2.4.4 Mesures de distance

L'algorithme PSOAlign spécifie que toutes les particules dans l'essaim se déplacer vers le chef (leader) de l'essaim à une vitesse proportionnelle à la distance entre chaque particule et le chef (leader). Ainsi, l'algorithme nécessite qu'une mesure de la distance entre les particules être spécifiée. La distance est donnée par le nombre de brèches qui ont les mêmes positions dans les deux alignements (particule, chef) divisé par le nombre totale des brèches ajoutées aux séquences de ces deux alignements.

$$distance = \frac{Nombre\ de\ brèches\ de\ memes\ positions}{Nombre\ total\ de\ brèches}$$

A titre d'exemple, soient les deux matrices suivantes qui décrivent deux alignements par les positions des brèches dans ses séquences :



Dans cet exemple, il claire que le nombre total des brèches est 18 (9 à partir de chaque particule), alors que le nombre de brèches qui ont les mêmes positions dans les deux alignements est 4, donc la distance est $\frac{18-4}{18} = 0.77$. Notez bien ici que la position 5 dans la première séquence du premier alignement désigne la première brèche ajoutée à cette séquence. Par contre cette même position désigne la deuxième brèche de la même séquence du deuxième alignement. Pour cette raison ces deux brèches ne sont pas confondues. La même décision est prise pour les brèches des position 5 et 6 de la deuxième séquence.

2.4.5 Les opérateurs

Dans l'algorithme PSO chaque particule doit déplacer vers le chef de l'essaim avec une vitesse proportionnelle à la distance entre la particule et le chef. Dans notre travail et pour interpréter ce mouvement, un opérateur similaire à l'opérateur de croisement utilisé par les algorithmes génétiques est proposé. L'idée de cet opérateur est comme suit : on choisit un point de croisement aléatoire *L* en fonction de la distance mesurée entre le chef et la particule (*L* doit être $\in$ [1, *distance* * *taille de la plus longue séquence*]) pour diviser l'alignement en deux segments, puis un segment de la particule est remplacé par un segment du chef. Ce remplacement consiste à changer les positions des brèches dans les segments en respectant l'ordre des caractères dans les séquences. L'opération de croisement sur l'exemple précédent et illustrée dans la figure 3.5 ci-dessous :Les deux alignements correspondants à l'exemple précédent sont :

Chef :

W	G	K	V	-	-	N	V	D
W	D	K	V	-	-	N	-	-
S	-	K	V	G	G	N	-	-

Particule :

W	G	K	-	-	V	N	V	D
-	W	D	K	-	-	-	V	N
S	-	K	V	G	G	-	N	-

Figure 3.5 : Exemple de mouvement d'une particule.

La distance entre la particule et le leader est 0.77 et la longueur de particule est 9. Ainsi le point de croisement doit être entre 1 et 0.77*9=7. En suppose que cette valeur est prise égale à 5. Le résultat de croisement entre le chef et la particule devient :

Résultat:

W	G	K	V	-	N	V	D
W	D	K	V	-	-	-	N
S	-	K	V	G	G	-	N

Notons que en certains cas, le caractère final de toutes les séquences dans un alignement devient une brèche ce qui génère une colonne de brèches, cette colonne est enlevée pour réduire le coût de pénalité pendant le calcul du score de l'alignement.

2.4.6 Critère d'arrêt

Plusieurs critères d'arrêt sont possibles pendant l'exécution des méthodes itérative, mais le critère le plus utilisé est le nombre d'itérations. Dans notre travail, nous avons choisi ce critère et l'exécution de notre algorithme peut s'arrêter après un nombre d'itérations *Itmax* déterminé par l'utilisateur.

3. Optimisation par algorithmes génétiques

3.1. Algorithmes génétiques [43]

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils ont été adaptés à l'optimisation par John Holland, également les travaux de David Goldberg ont largement contribué à les enrichir Le vocabulaire utilisé est le même que celui de la théorie de l'évolution et de la génétique, on emploie le terme individu (solution potentielle), population ensemble de solutions),

génotype (une représentation de la solution), gène (une partie du génotype), parent, enfant, Reproduction, croisement, mutation, génération, etc.

Leur fonctionnement est extrêmement simple, on part d'une population de solutions Potentielles (chromosomes) initiales, arbitrairement choisies.

On évalue leur performance (Fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. Quelques individus se reproduisent, d'autres disparaissent et seuls les individus les mieux adaptés sont supposés survivre. On recommence ce cycle jusqu'à ce qu'on trouve une solution satisfaisante. En effet, l'héritage génétique à travers les générations permet à la population d'être adaptée et donc répondre au critère d'optimisation, la figure 3.6 illustre les principales étapes d'un algorithme génétique.

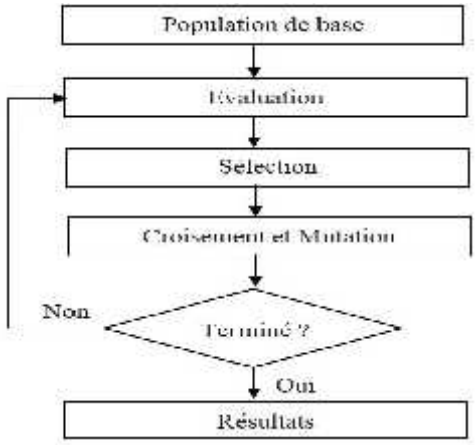


Figure 3.6: Fonctionnement d'un algorithme génétique

Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Son mise en œuvre nécessite :

3.1.1 Le codage de données

La première étape est de définir et coder convenablement le problème. Cette étape associe à chaque point de l'espace de recherche une structure de données spécifique, appelée génotype ou ensemble de chromosomes, qui caractérisera chaque individu de la population. Le codage de chaque individu en séquence est essentielle dans l'élaboration d'un algorithme génétique dont dépend notamment l'implémentation des opérateurs de transformations. Ainsi, cette phase détermine la structure de données qui sera utilisée pour coder le génotype des individus de la population. Le codage doit donc être adapté au problème traité.

Plusieurs types de codages sont utilisés dans la littérature, les premiers résultats théoriques sur les algorithmes génétiques ont opté pour un codage par une séquence binaire de longueur fixe à travers la notion de schéma. L'efficacité de l'algorithme génétique dépend donc du choix convenable du type de codage.

3.1.2 Génération de la population initiale

La génération de la population initiale, c'est-à-dire le choix des dispositifs de départ que nous allons faire évoluer, ce choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme.

Néanmoins, une initialisation aléatoire est plus simple à réaliser : les valeurs des gènes sont tirées au hasard selon une distribution uniforme. Toutefois, il peut être utile de guider la génération initiale vers des sous domaines intéressants de l'espace de recherche.

Par exemple lors d'une recherche d'optima dans un problème d'optimisation sous contraintes, il est préférable de produire des éléments satisfaisant les contraintes. La population initiale doit être suffisamment diversifiée et de taille assez importante pour que la recherche puisse parcourir l'espace d'état dans un temps limité.

3.1.3 Fonction d'adaptation (Fitness)

L'évaluation de la Fitness est généralement l'étape dans laquelle on mesure la performance de chaque individu. Pour pouvoir juger la qualité d'un individu et ainsi le comparer aux autres, il faut établir une mesure commune d'évaluation. Aucune règle n'existe pour définir cette fonction, son calcul peut ainsi être quelconque, que ce soit une simple équation ou une fonction affine. La manière la plus simple est de poser la fonction d'adaptation comme la formalisation du critère d'optimisation.

3.1.4 Sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais, pendant le passage d'une génération à une autre, ce processus est basé sur la performance de l'individu. L'opérateur de sélection doit être conçu pour donner également une chance aux mauvais éléments, car ces éléments peuvent, par croisement ou mutation, engendrer une descendance pertinente par rapport au critère d'optimisation. Il existe différentes techniques de sélection par rapport la sélection (uniforme, par tournoi, élitisme, roulette**)** .

- **Sélection uniforme :** On ne s'intéresse pas à la valeur d'adaptation fitness et la sélection s'effectue d'une manière aléatoire et uniforme telle que chaque individu *i a* la même probabilité $Prob(i)=1/T_{pop}$ comme tous les autres individus ($_{Tpop}$ est la taille de la population).
- **Sélection par tournoi :** Deux individus sont choisis au hasard, on compare leurs fonctions d'adaptation et le mieux adapté est sélectionné.
- **Elitisme :** Cette méthode de sélection permet de favoriser les meilleurs individus de la population. Ce sont donc les individus les plus prometteurs qui vont participer à l'amélioration de notre population. On peut constater que cette méthode induisait une convergence prématurée de l'algorithme.
- **Sélection par roulette :** La sélection des individus par la méthode de la roulette s'inspire de la roue de loterie sur laquelle chaque individu est représenté par un secteur proportionnel à sa fitness. On fait tourner la roue et on sélectionne un individu. Les individus les mieux évalués ont statistiquement plus de chance d'être sélectionnés, mais donne aussi une possibilité aux Individus mal adaptés d'être choisis. A chaque individu une probabilité est associée, d'être choisi proportionnelle à son adaptation f_i: $Prob(i)= f_i / \sum f_i$, où désigne la somme des adaptations de la population.
- **Reste stochastique sans remplacement :** Cette sélection associe la sélection par roulette et la sélection déterministe. Un nombre minimal de représentants de chaque

individu parmi les futurs parents est déterminé par avance en fonction de son adaptation, puis la population sera complétée par un tirage aléatoire. Pour un individu *i*, le nombre de représentation dans la future génération est donné par E(f_i), où E désigne la partie entière et fi désigne l'adaptation de *i* rapportée à la moyenne des adaptations de tous les individus. La génération suivante est alors complétée par la méthode de sélection par roulette telle que l'évaluation d'un individu est donnée par le reste stochastique *:* $r_i = f_i - E(f_i)$

3.1.5 Croisement

L'opérateur de croisement favorise l'exploration de l'espace de recherche et enrichit la diversité de la population en manipulant la structure des chromosomes, le croisement fait avec deux parents et génère deux enfants, en espérant qu'un des deux enfants au moins héritera de bons gènes des deux parents et sera mieux adapté qu'eux.

Il existe plusieurs méthodes de croisement par exemple le croisement en un point, ou en multiples points voir les figures 3.7 et 3.8.

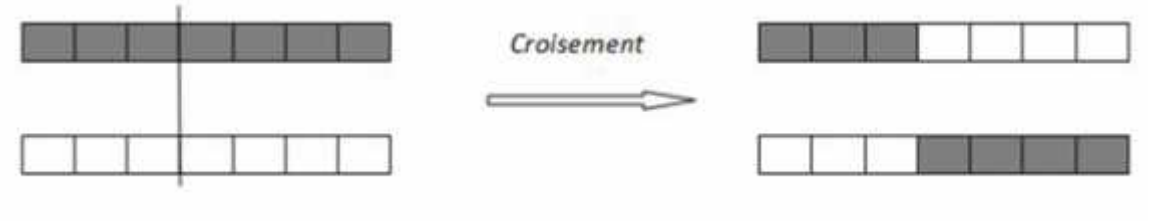


Figure 3.7: Croisement a un point.

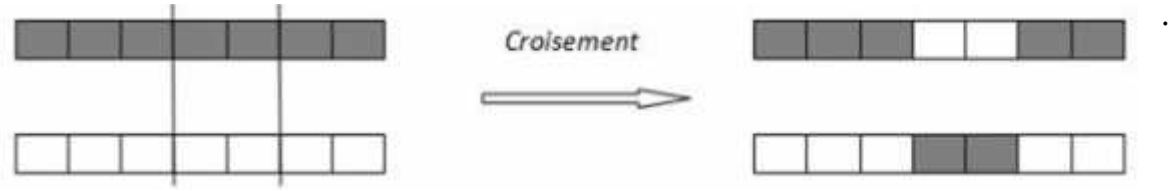


Figure 3.8: Croisement a deux points.

3.1.6 Mutation

L'opérateur de mutation est un processus où un changement mineur du code génétique appliqué à un individu pour introduire de la diversité et ainsi d'éviter de tomber dans des optimums locaux.

Cet opérateur est appliqué avec une probabilité Pm généralement inférieure à celle du croisement Pc, voir la figure 3.9

.

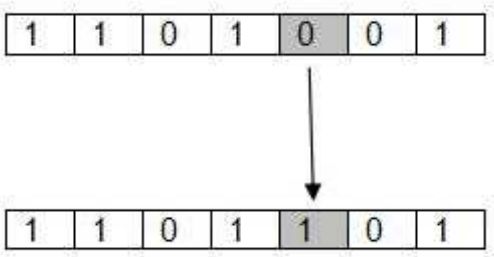


Figure 3.9: Représentation schématique d'une mutation dans le cas d'un codage binaire

L'efficacité des algorithmes génétiques dépend fortement du réglage des différents paramètres caractérisant ces algorithmes, et qui sont parfois difficiles à déterminer. Des paramètres comme la taille de la population, le nombre maximal des générations, la probabilité de mutation pm, et la probabilité de croisement pc.

Les deux premiers paramètres dépendent directement de la nature du problème et de sa complexité, et leurs choix doit représenter un compromis entre la qualité des solutions et le temps d'exécution. La probabilité de croisement pc est liée à la forme de la fonction dévaluation. Son choix est en général heuristique. Plus sa valeur est élevée, plus la population subit des changements importants.

La probabilité de mutation pm est généralement faible puisqu'un taux élevé risque de conduire vers un optimum local. En revanche, une probabilité faible permet d'assurer une bonne exploration de l'espace de recherche sans perturber la convergence. Le succès des algorithmes génétiques dépend aussi de la manière du codage des individus.

Dans les problèmes combinatoires, ce codage est souvent suggéré par la nature même du problème, ce qui peut induire de meilleurs résultats.

3.2. Application de l'algorithme génétique au problème MSA

Dans cette section on veut détailler les différents éléments de l'algorithme génétique choisie pour la résolution du problème MSA, en basant sur les éléments de base décrits concernant l'algorithme générale des algorithmes génétiques.

La version de l'algorithme génétique proposé pour résoudre le problème MSA est notée dans le reste de manuscrit par GAAlign, elle est résumée dans la table 3.2 ci-dessous :

Table 3.3 : Pseudo-code de GAAlign

Début
Initialiser P_{size} ,P_{mut} ,*nbr_iteration*,$\mu = 0$;
Générer Aléatoirement P_{size} Alignements (population) **;**
Trier la population par ordre décroissant
Répéter
i=0 ;
Tant que($\mu < P_{size}/2$) **faire**
Sélectionner deux parentes (*u, u+1*) ;
Faire le croisement du deux parents sélectionnés pour obtenir deux enfants
Muter les deux enfants par une probabilité P_{mut}
u= u+2 ;
Fin tant que
Calculer la fonction *fitness* pour chaque chromosome
Ranger les parents et les enfants dans l'ordre décroissant selon leur fonction d'évaluation
Remplacer *(P_{size} , $Best_{pop}$) ;*
i++ ;
Jusqu'à (i < *nbr_iteration*) ;
Enregistrer la fitness du meilleur chromosome
Fin

3.2.1 Codage de solution

Le codage de solution est une étape très importante pour un algorithme génétique. Dans notre travail nous avons choisi la technique la plus simple : chaque séquence est représentée par la suite des caractères qui la composent et ainsi les brèches ajoutées. Par conséquent, un chromosome est représenté par un alignement admissible composé par un ensemble de séquences.

3.2.2 Génération de la population initiale

La génération de la population initiale, c'est-à-dire le choix des dispositifs de départ que nous allons faire évoluer, ce choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme

Pour notre travail, et à cause de sa simplicité, une initialisation aléatoire de cette population est adoptée. Donc, les l'ensemble des brèches ajoutées aux différentes séquences sont tirées au hasard dans l'intervalle [*1, 1.2*la taille de la plus longue séquence]*.

3.2.3 Fonction d'adaptation « Fitness »

La fonction *fitness* « *d'adaptation* » *F(x)*, mesure la puissance de chaque chromosome à s'adapter, elle décrit l'aptitude d'un chromosome de la population, à optimiser l'objectif. Dans notre travail et de raison de simplicité et comme nous somme dans le cas de maximisation, nous avons choisi la fonction de score décrite dans la section 2.4.3. Alors la fonction d'adaptation retenue est la même utilisée pour la méthode PSOAlign.

3.2.4 Sélection

La sélection permet d'identifier les individus susceptibles dans une population d'être croisés dans une population. Plusieurs techniques de sélection sont décrites dans la section 3.1.4. Dans notre étude, nous avons choisi une méthode de sélection qui prend à chaque fois deux chromosomes : un chromosome *i* et son chromosome suivant *(i+1)* telle que ces deux chromosomes sont ordonnées par ordre décroissant de leurs fonction d'évaluation (fitness).

3.2.5 Croisement

L'opérateur de croisement est appliqué sur deux ou plusieurs chromosomes parents pour obtenir deux ou plusieurs chromosomes enfants. Pour notre algorithme nous somme opté pour le croisement a un point à cause de sa simplicité et sa rapidité. Il consiste à diviser chacun des deux parents en deux parties à la même position, choisie au hasard dans l'intervalle [*1, 1.2*la taille de la plus longue séquence*] et à recopier la partie inférieure du parent à l'enfant et à compléter les gènes manquants de l'enfant à partir de l'autre parent en maintenant l'ordre des gènes. Cette technique nous permet de concevoir un chromosome valide. Un exemple de ce type de croisement est illustré dans la figure 3.10 ci-dessous :

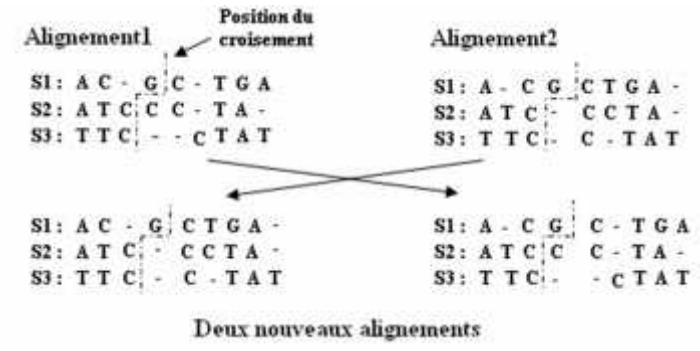


Figure 3.10 : Le croisement entre deux parents

3.2.6 Mutation

La mutation apporte l'aléa nécessaire à une exploration efficace de l'espace de solutions. Elle permet de quitter les extrêmes locaux. Cet opérateur de mutation est utilisé

avec une probabilité P_{mut} . Si β généré aléatoirement, appartient à [0,P_{mut}], nous appliquons l'opérateur de mutation sur cet individu. La technique retenue pour notre étude est comme suit : Selon la probabilité de mutation, on choisit aléatoirement une séquence de l'alignement objet de la mutation puis on sélectionne aléatoirement une brèche suivie/précédée par un caractère et on fait la permutation entre eux. Cette opération est présentée dans la figure 3.11 suivante :

Figure 3.11 : Illustration d'une mutation sans une séquence

3.2.7 Critère d'arrêt

Le critère d'arrêt utilisé est un nombre d'itérations *Itmax*. L'algorithme génétique s'arrête et donne le meilleur chromosome qui possède le meilleur score. Dans notre algorithme nous avons essayé de prendre un nombre *Itmax* qui respecte les contraintes du problème traité.

4. Conclusion

Dans ce chapitre, et en premier temps nous avons présenté en détail les deux métaheuristiques : l'algorithme PSO et l'algorithme génétique GA, ses principes et ses paramètres. En deuxième temps nous avons adapté chacune d'elle pour résoudre le problème d'alignement multiple de séquences de protéines. Les deux versions adaptées après la détermination de l'ensemble des paramètres sont notées PSOAlign et GAAlign. Le chapitre suivant sera consacré à la réalisation et les résultats expérimentaux de notre étude.

CHAPITRE IV

Réalisation et expérimentation

1. Introduction

La réalisation est la dernière phase dans tout processus de développement d'un système ou d'un logiciel. Dans ce chapitre, on vise de présenter brièvement les outils et les moyens utilisés pour implémenter nos méthodes proposées dans le chapitre précédent. En particulier, la démarche de conception retenue, l'environnement de programmation choisi, et l'ensemble des interfaces générés par notre application.

2. Environnement de développement

Le système d'exploitation choisi pour la réalisation de notre application est le système Windows 7 Professionnel c'est un système connu pour son efficacité, sa fiabilité, sa robustesse, et sa sécurité ainsi que la richesse de ses outils de programmations.

Nous avons choisi le langage C# pour développer notre Systèm. Ce choix de langage est motivé par les raisons suivantes :

- Le langage *C#* permet l'utilisation des classes d'objets et d'appliquer par conséquence les techniques avancées de la programmation orientée objet.
- Les compilateurs C# sont actuellement implémentés sur toutes les plates-formes Windows, ce qui a fait du langage C# un outil de programmation très répandu.
- Le code généré par le compilateur C# est très optimisé, ce qui rend les exécutables plus compactes et plus rapides.
- La plupart des implémentations des algorithmes standards sont implémentés à bas de langage C#.

Nous avons exploité l'environnement de programmation ***Microsoft Visuale studio 2012 (c#)***, et utilisé l'environnement ***Windows Forms*** pour la réalisation de l'interface graphique.

3. L'architecture de l'application

Nous avons opté pour la programmation orienté objet (POO) par ce qu'elle permet une meilleur lisibilité du code source et une simplicité considérable du débogage et de la détection des erreurs, la réutilisation, la modularité et la facilité de maintenance …etc.

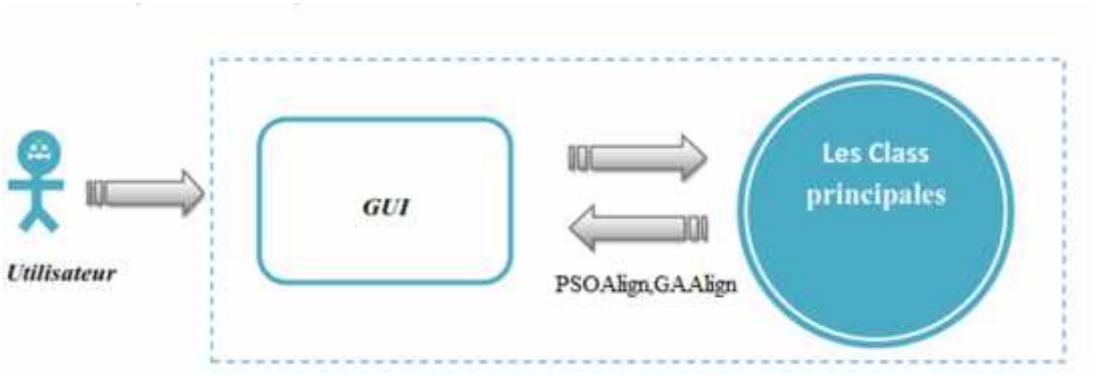


Figure 4.1 L'architecture de l'application

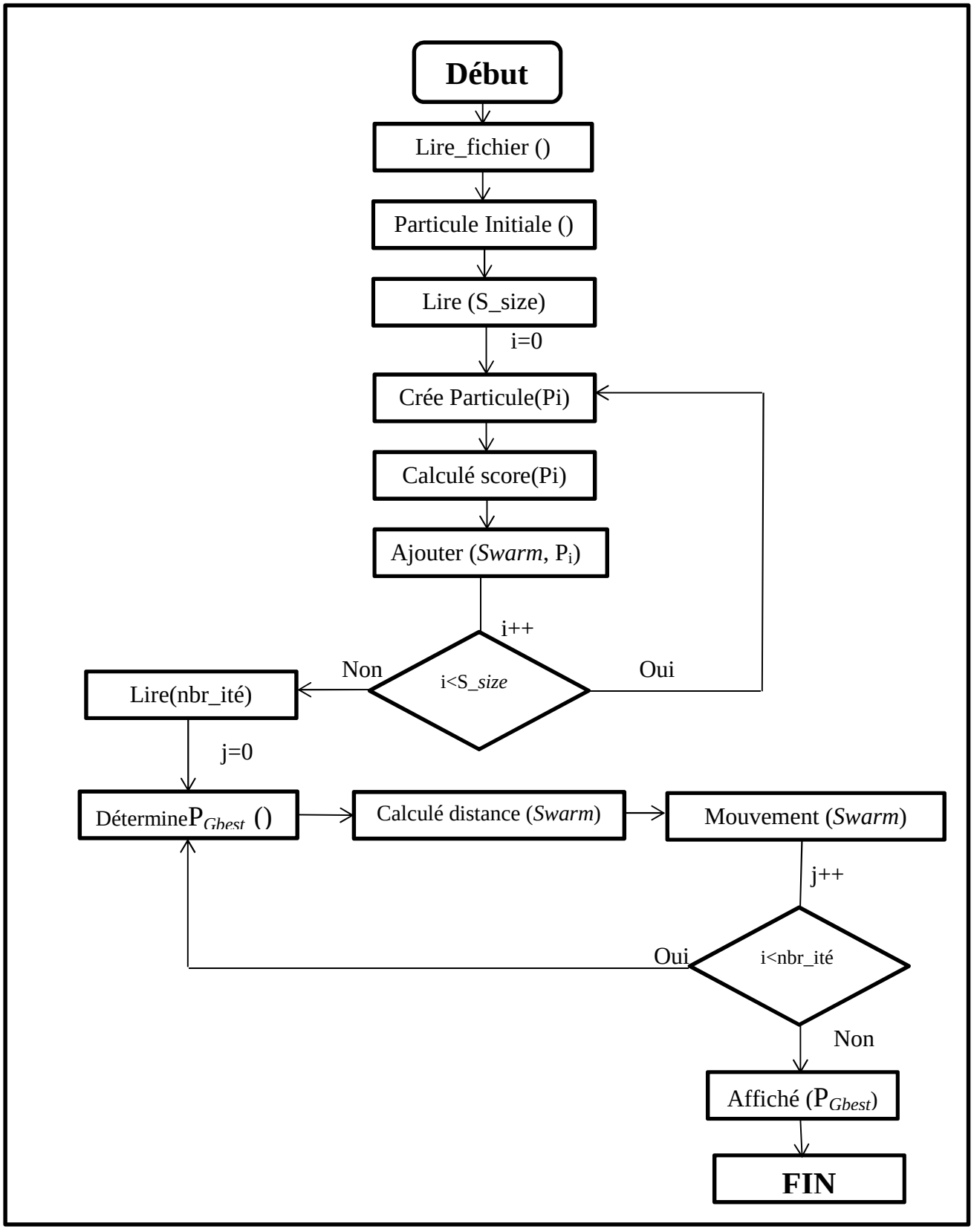


Figure 4.2 Organigramme de notre système (PSOAlign)

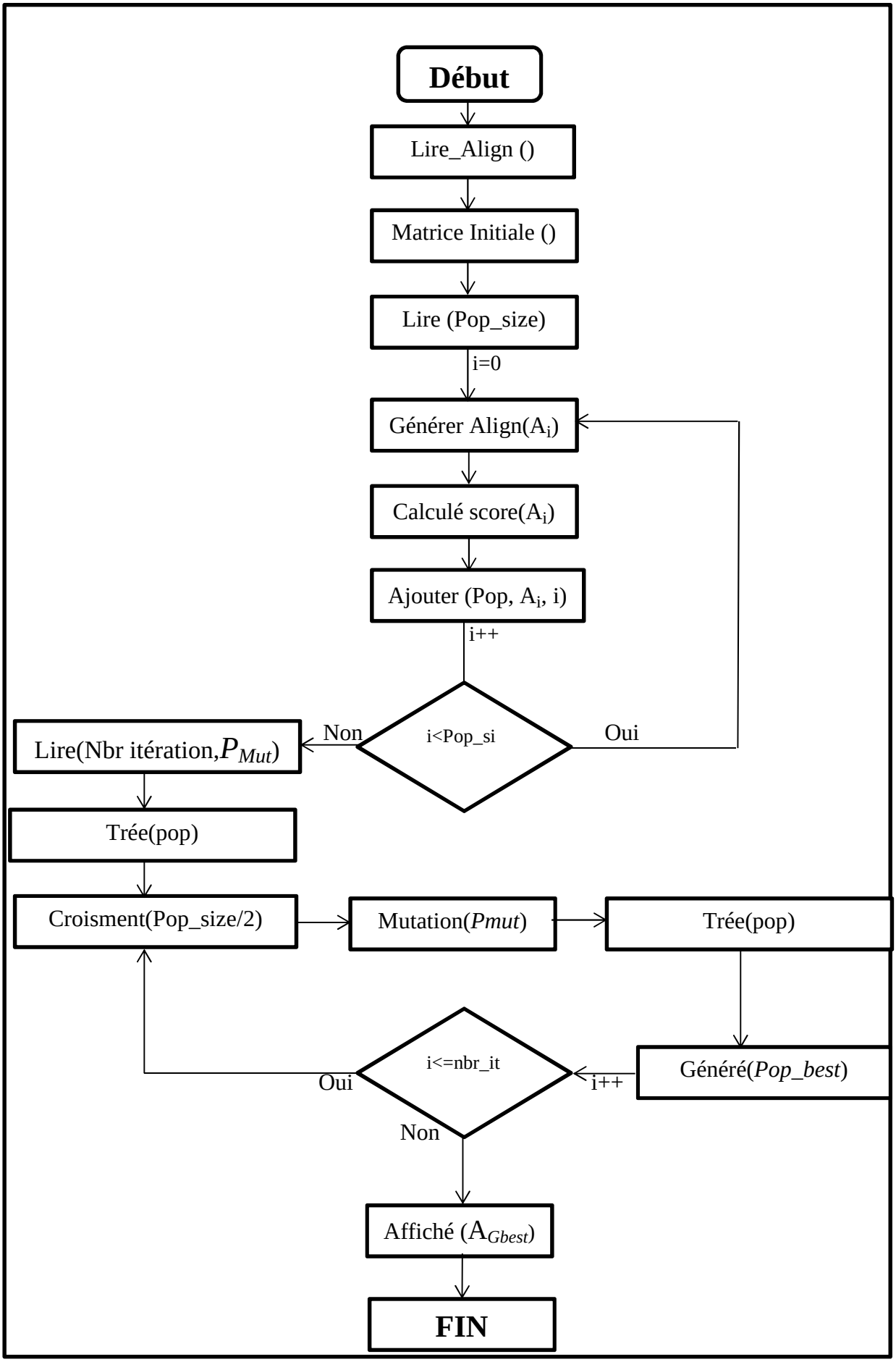


Figure 4.3 Organigramme de notre système (GAAlign)

3.1. GUI (Graphic User Interface)

C'est une interface graphique simple qui permet aux utilisateurs d'interagir avec l'application. Même si les algorithmes ne sont pas destinés au grand public, nous avons essayé de donner un aspect visuel, attrayant et convivial à notre application à travers une interface graphique ergonomique.

3.2. Les Classes principales

Nous avons pu ressortir les classes utilisées tout au long de la programmation. L'ensemble des classes et leurs méthodes respectives sont présentées dans les figures suivants :

Les classes générées dans l'algorithme PSOAlign sont :

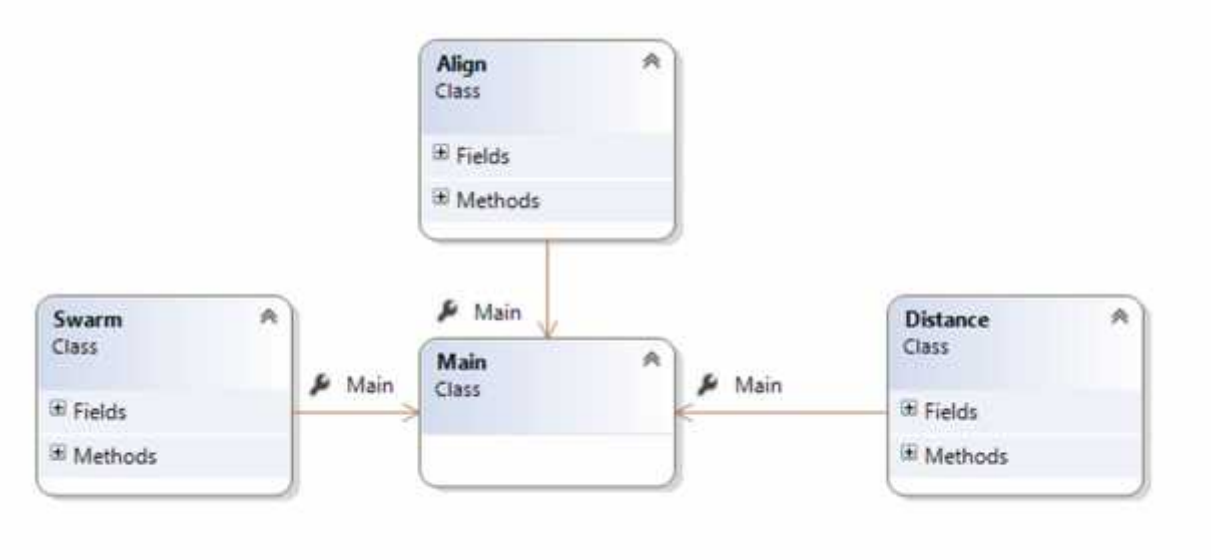


Figure 4.5 les classes principales de PSOAlign

Les classes générées dans l'algorithme GAAlign sont :

Figure 4.6 les classes principales de GAAlign

4. Résultats expérimentaux

Dans ce qui suit, nous allons présenter les différents tests et évaluations effectués ainsi que les résultats obtenus par les algorithmes proposés pour prouver leurs importances d'un point de vue de la recherche d'une solution approchée au problème sujet d'étude. Ensuite et afin d'évaluer la performance des algorithmes, une comparative est menée entre nos deux algorithmes sur des instances de différentes tailles de données biologiques de *Proteines***.** Les conditions de l'environnement nous avons exploité une machine dotée d'un microprocesseur *Intel® Core i5-2450M CPU@ 2.5GHz*, d'une RAM de 4Go et fonctionnant sous le système d'exploitation Windows 7 32 bit.

4.1. Données utilisées

Pour faire les tests de nos algorithmes on utilise la base de données biologiques '*BaliBase'* qui contient plusieurs types de données sous forme des alignements avec leurs alignements de référence. Pour nos tests, nous avons choisi un ensemble d'alignements de tailles courtes, moyennes et longues avec leurs identifications. Les caractéristiques des données utilisées sont regroupées dans le table 4.1 ci-dessous.

Référence	Nom	Longueur	Identification %
1ubi	Ubiquitin	Short	<25
1aab	high mobility group protein	Short	20-40
1csp	cold shock protein	Short	>35
1hav A	hepatitis proteinase	Medium	<25
1ton	tonin	Medium	20-40
1ppn	papain	Medium	>35
1pam A	cyclodextrin	Large	<25
1ac5	carboxypeptidase	Large	20-40

Table 4.1 les données utilisé pour les testes

4.2. Critères d'évaluation

Pour évaluer les performances de chaque 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 :

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

4.3. Résultats obtenus

4.3.1. Pour PSOAlign

Référance				Résultat			
Nom	Nombre séquence	séquence Max	Séquence Min	Score Initiale	Score Final	Taux d'amélioration	temps d'exécution(S)
1ubi	4	94	76	-1100	-934	15,09%	2
1aab	4	79	67	-716	-566	20,94%	1
1csp	5	70	66	-699	-400	42,78%	2
1hav A	5	199	136	-3249	-2840	13,40%	7
1ton	5	244	224	-3569	-3246	9,05%	10
1ppn	5	220	212	-2449	-2180	10,98%	8
1pam A	5	572	435	-10002	-9578	4,24%	43
1ac5	4	483	421	5832	-5400	7,41%	19

Table 4.2 Résultats obtenus pour (*swarm_size*=30, *nbr_itération*=100)

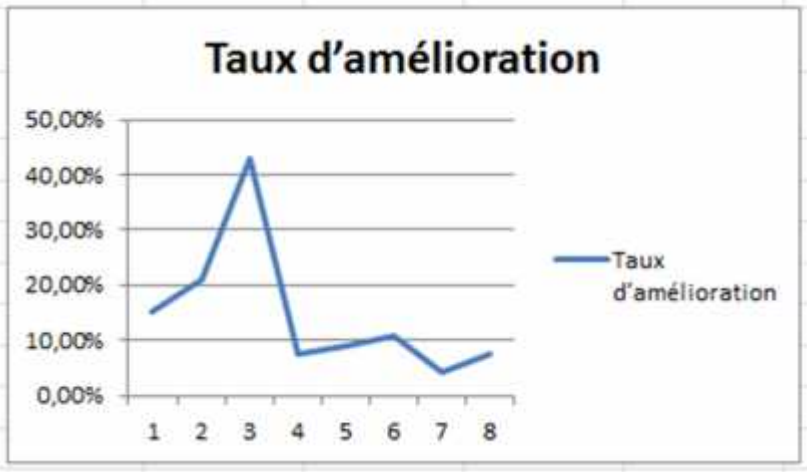


Figure 4.7 Taux d'amélioration de score pour (*swarm_size*=30, *nbr_itération*=100)

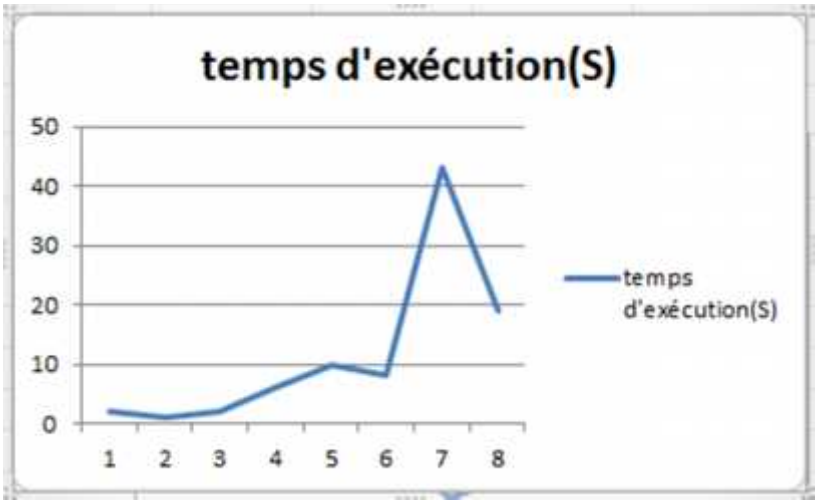


Figure 4.8 Temps d'exécution pour (*swarm_size*=30, *nbr_itération*=100)

Référance				Résultat			
Nom	Nombre séquence	séquence Max	Séquence Min	Score Initiale	Score Final	Taux d'amélioration	temps d'exécution(S)
1ubi	4	94	76	-1090	-924	15,26%	7,66
1aab	4	79	67	-687	-535	22,03%	7
1csp	5	70	66	-632	-285	54,88%	10
1hav A	5	199	136	-3150	-2860	9,21%	27
1ton	5	244	224	-3481	-2908	16,47%	36
1ppn	5	220	212	-2543	-2141	15,79%	32
1pam A	5	572	435	-10012	-9362	6,49%	38
1ac5	4	483	421	-5762	-5220	9,41%	47

Table 4.3 Résultats obtenus pour (*swarm_size*=100, *nbr_itération*=200)

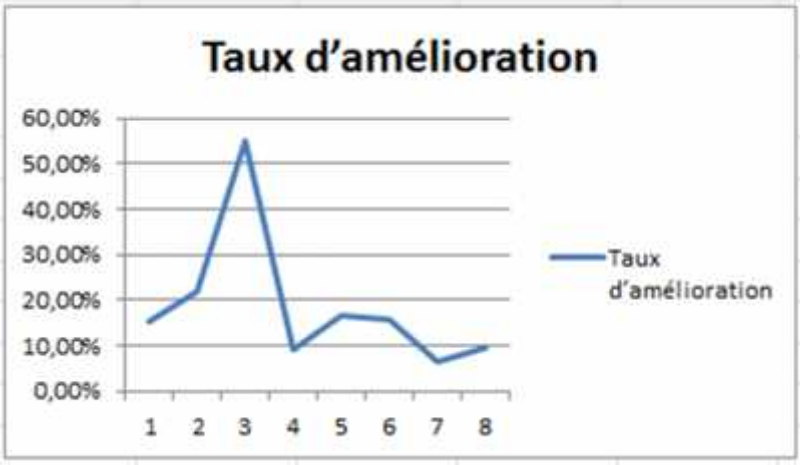


Figure 4.9 Taux d'amélioration de score pour (*swarm_size*=100, *nbr_itération*=200)

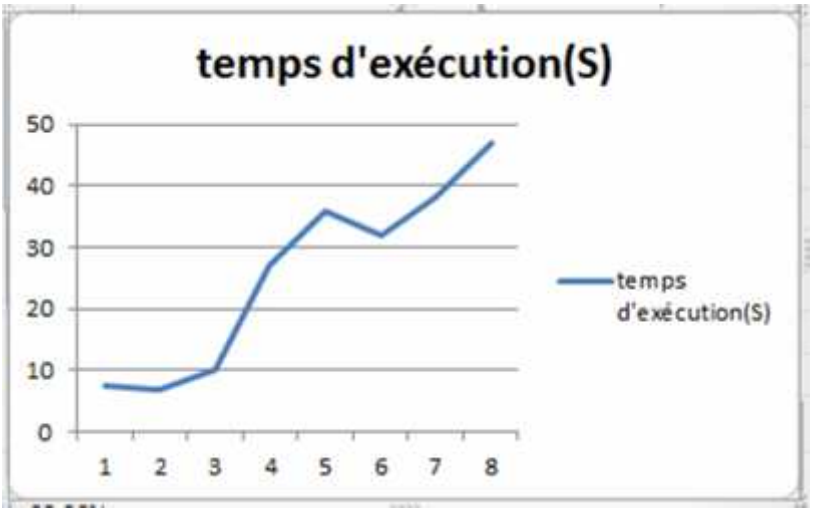


Figure 4.10 Temps d'exécution pour (*swarm_size*=100, *nbr_itération*=200)

Référance				Résultat			
Nom	Nombre séquence	séquence Max	Séquence Min	Score Initiale	Score Final	Taux d'amélioration	temps d'exécution(S)
1ubi	4	94	76	-1067	-898	15,81%	35
1aab	4	79	67	-672	-475	29,25%	35
1csp	5	70	66	-672	-66	90,18%	59
1hav A	5	199	136	-3066	-2822	7,96%	150
1ton	5	244	224	-3366	-3000	10,86%	117
1ppn	5	220	212	-2342	-1816	22,46%	78
1pam A	5	572	435	-10012	-9362	6,49%	38
1ac5	4	483	421	-5824	-5008	14,01%	137

Table 4.4 Résultats obtenus pour (*swarm_size*=250, *nbr_itération*=500)

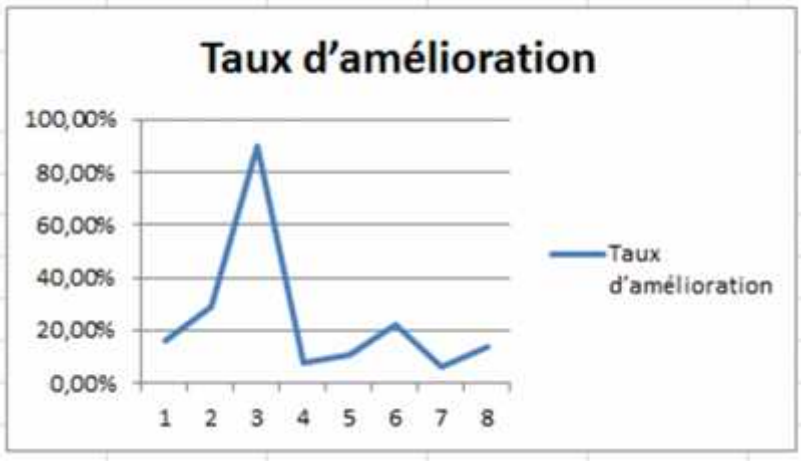


Figure 4.11 Taux d'amélioration de score pour (*swarm_size*=250, *nbr_itération*=500)

Figure 4.12 Temps d'exécution pour (*swarm_size*=250, *nbr_itération*=500)

Réf	Particules	Itération	Particules	Itération	Particules	Itération
	30	100	100	200	250	500
1ubi	-934	15,09%	-924	15,26%	-898	15,81%
1aab	-566	20,94%	-535	22,03%	-475	29,25%
1csp	-400	42,78%	-285	54,88%	-66	90,18%
1hav A	-2990	7,68%	-2860	9,21%	-2822	7,97%
1ton	-3246	9,05%	-2908	16,47%	-3000	10,86%
1ppn	-2180	10,98%	-2141	15,79%	-1816	22,46%
1pam A	-9578	4,24%	-9472	5,11%	-9362	6,49%
1ac5	-5400	7,41%	-5220	9,41%	-5008	14,01%

Table 4.5 Résultats d'évolution de score pour PSOAlign

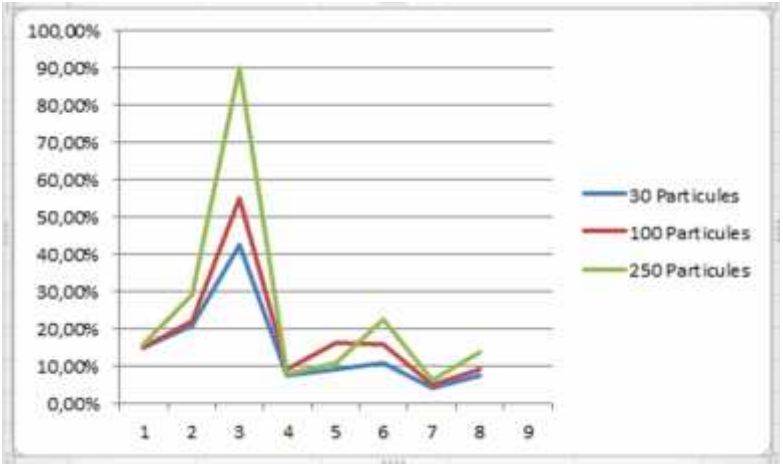


Figure 4.13 Taux d'amélioration de score global pour PSOAlign de chaque itération.

❖ **Analyse du résultat pour PSOAlign**

L'analyse des résultats présentés dans les tables 4.2, 4.3 et 4.4 et les figures 4.7 à 4.13 montrent clairement que la méthode PSOAlign améliore grandement la solution initiale. En effet, le taux d'amélioration du score dans le mauvais cas est 4% mais il atteint dans certain cas 90%. Cette amélioration dépend aussi de la taille de la population initiale générée, dont les meilleurs résultats sont donnés par un essaim composé de 250 particules. Le temps d'exécution est largement lié au nombre d'itérations et la taille de l'essaim. En effet, il est très raisonnable pour les instances de courtes et moyennes tailles, alors qu'il augmente progressivement quand il s'agit des instances de grandes tailles.

4.3.2. Pour GAAlign

Référance				Résultat			
Nom	Nombre séquence	séquence Max	Séquence Min	Score Initiale	Score Final	Taux d'amélioration	temps d'exécution(S)
1ubi	4	94	76	-1152	-998		4
1aab	4	79	67	-690	-630	8,70%	3
1csp	5	70	66	-699	-590	15,59%	7
1hav A	5	199	136	-3145	-3050	3,02%	10
1ton	5	244	224	-3421	-3395	0,76%	11
1ppn	5	220	212	-2498	-2250	9,93%	11
1pam A	5	572	435	-10123	-9802	3,17%	47
1ac5	4	483	421	-5802	-5620	3,14%	23

Table 4.6 Résultats obtenus pour (*pop_size*=30, *nbr_itération*=100, *Pmut*=0,1)

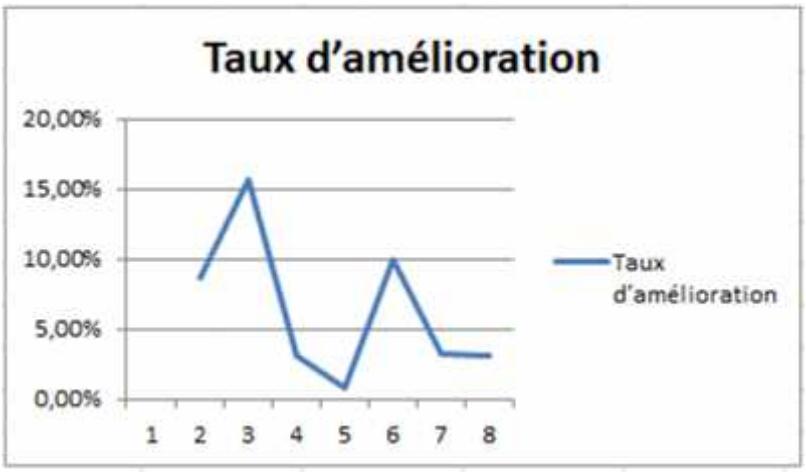


Figure 4.14 Taux d'amélioration de score pour (*Pop_size*=30, *nbr_itération*=100,*Pmut*=0,1)

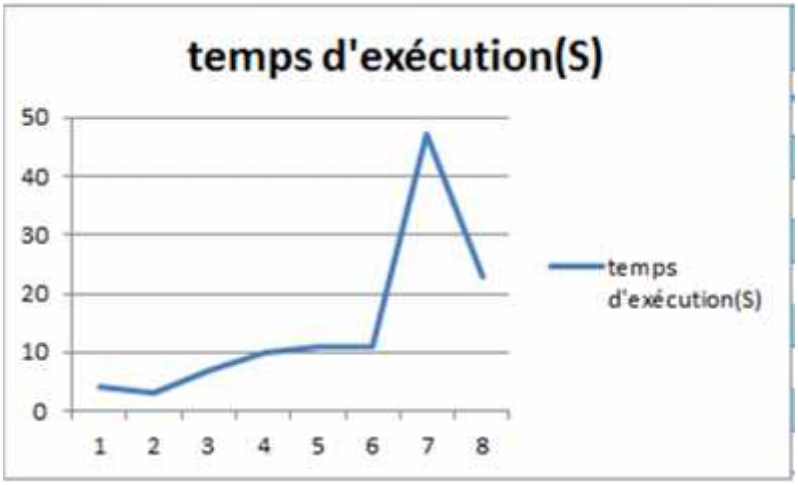


Figure 4.15 Temps d'exécution pour (*Pop_size*=30, *nbr_itération*=100, *Pmut*=0,1)

Référance				Résultat			
Nom	Nombre séquence	séquence Max	Séquence Min	Score Initiale	Score Final	Taux d'amélioration	temps d'exécution(S)
1ubi	4	94	76	-1032	-914	11,43%	9
1aab	4	79	67	-699	-598	14,45%	11
1csp	5	70	66	-632	-326	48,42%	13
1hav A	5	199	136	-3204	-3002	6,30%	25
1ton	5	244	224	-3462	-2960	14,50%	33
1ppn	5	220	212	-2493	-2265	9,15%	41
1pam A	5	572	435	-9921	-9658	2,65%	322
1ac5	4	483	421	-5802	-5432	6,38%	57

Table 4.7 Résultats obtenus pour (*pop_size*=100, *nbr_itération*=200, *Pmut*=0,1)

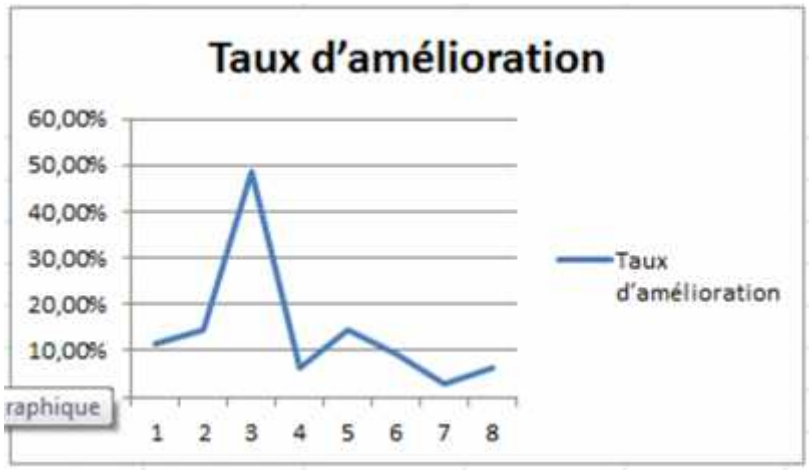


Figure 4.16 Taux d'amélioration de score pour (*pop_size*=100, *nbr_itération*=200, *Pmut*=0,1)

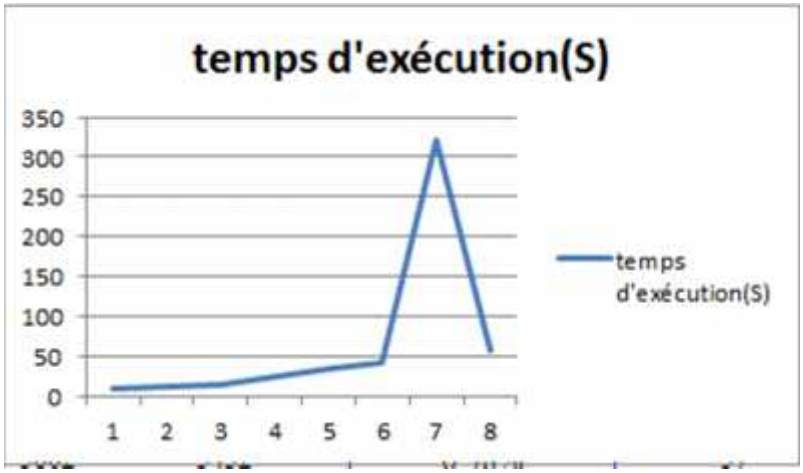


Figure 4.17 Temps d'exécution pour (*pop_size*=100, *nbr_itération*=200, *Pmut*=0,1)

Référance				Résultat			
Nom	Nombre séquence	séquence Max	Séquence Min	Score Initiale	Score Final	Taux d'amélioration	temps d'exécution(S)
1ubi	4	94	76	-1052	-922	12,36%	35
1aab	4	79	67	-521	-432	17,08%	35
1csp	5	70	66	-652	-255	60,89%	59
1hav A	5	199	136	-3014	-2803	7,00%	150
1ton	5	244	224	-3296	-2903	11,92%	117
1ppn	5	220	212	-2451	-2035	16,97%	78
1pam A	5	572	435	-9871	-9032	8,50%	38
1ac5	4	483	421	-5798	-5236	9,69%	137

Table 4.8 Résultats obtenus pour (*pop_size*=250, *nbr_itération*=500, *Pmut*=0,1)

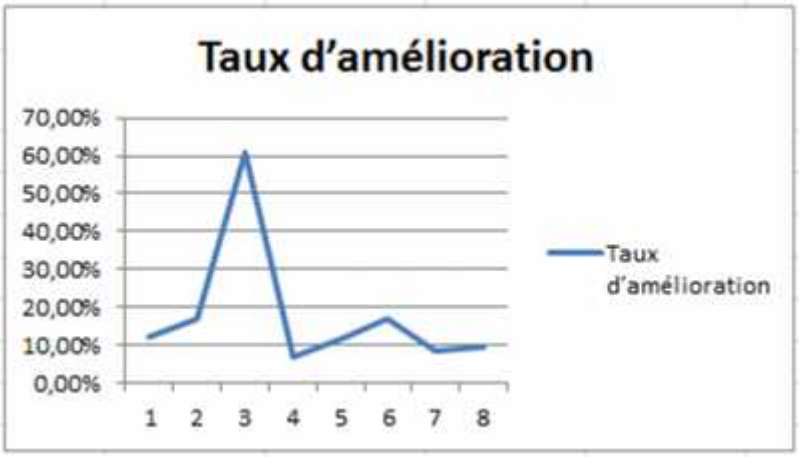


Figure 4.18 Taux d'amélioration de score pour (*Pop_size*=250, *nbr_itération*=500,*Pmut*=0,1)

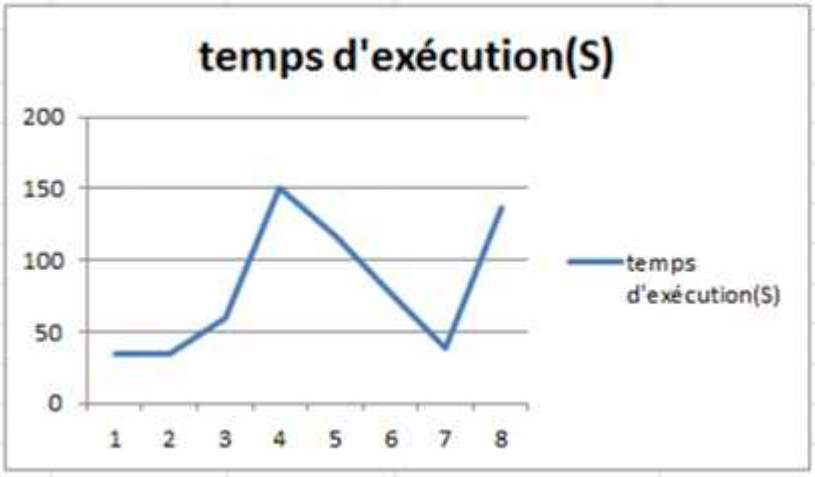


Figure 4.19 Temps d'exécution pour (*Pop_size*=250, *nbr_itération*=500, *Pmut*=0,1)

Réf	Pop_size	Itération	Particules	Itération	Pop_size	Itération
	30	100	100	200	250	250
1ubi	-998	8,68%	-914	11,43%	-922	12,36%
1aab	-630	8,70%	-598	14,45%	-501	17,08%
1csp	-590	15,59%	-326	48,42%	-255	60,89%
1hav A	-3050	3,02%	-3002	6,30%	-2903	7,00%
1ton	-3395	0,76%	-2960	14,50%	-2903	11,92%
1ppn	-2250	3,93%	-2265	9,15%	-2035	16,97%
1pam A	-9802	3,17%	-9658	2,65%	-9032	8,50%
1ac5	-5620	3,14%	-5432	6,38%	-5236	9,69%

Table 4.9 Résultats d'évolution de score pour GAAlign

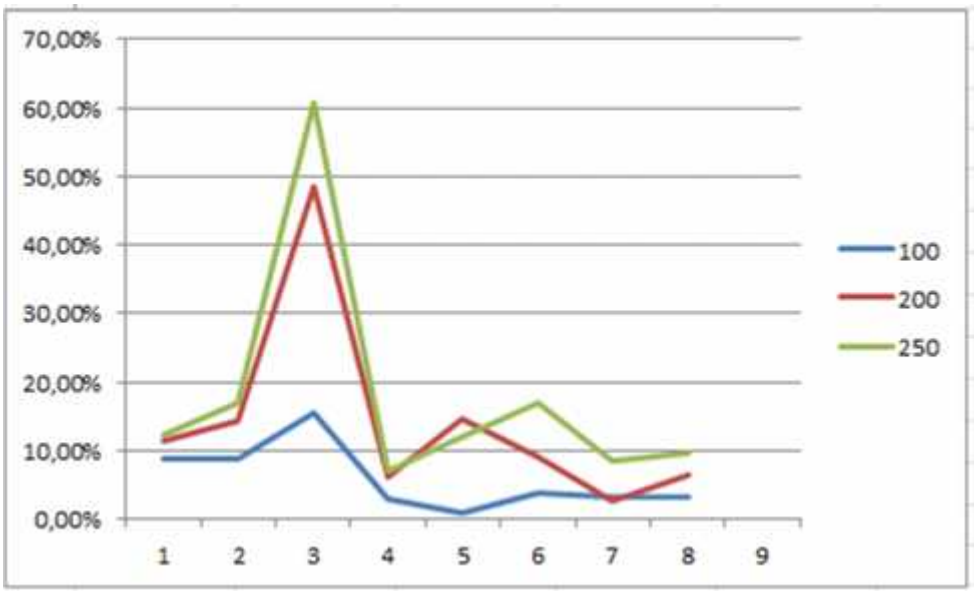


Figure 4.20 Taux d'amélioration de score global pour GAAlign de chaque itération.

❖ **Analyse du résultat pour GAAlign**

L'analyse des résultats présentés dans les tables 4.6 4.7 et 4.8 et les figures 4.14 à 4.19 montrent clairement que la méthode GAAlign améliore grandement la solution initiale. Cette amélioration dépend aussi de la taille de la population initiale générée, Le temps d'exécution est largement lié au nombre d'itérations et la taille de population.

Etude Comparative entre PSOAlign et GAAlign

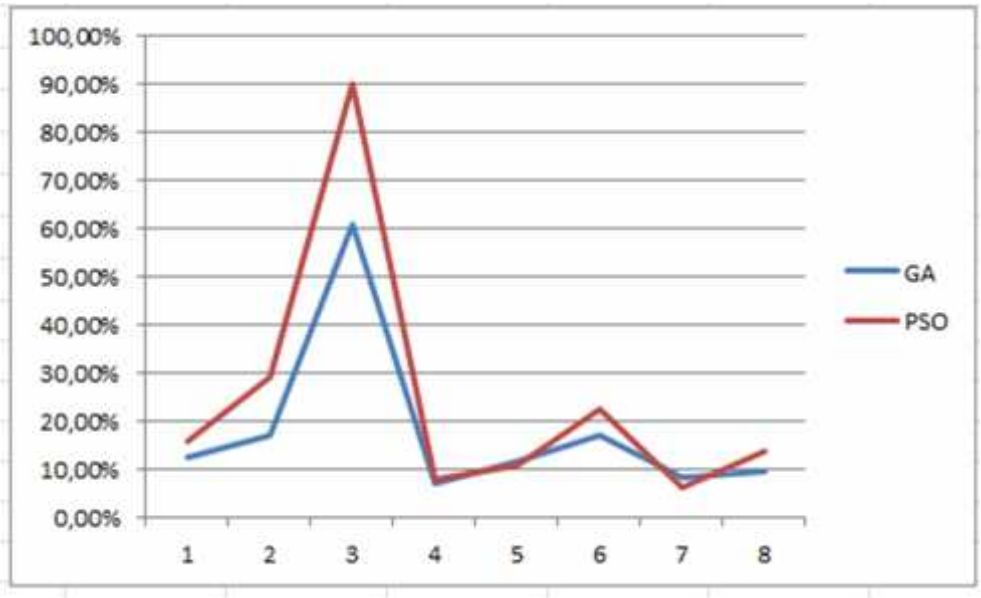


Figure 4.21 taux d'amélioration en cas de (nbr_itér=500,pop_size=250) on deux algorithmes (PSOAlign,GAAlign)

4.4. Les Interfaces du logiciel développé

- L'interface principale de notre système, à partir de cette page peut accéder à l'un de algorithmes implémente (PSOAlign, GAAlign) ou calculé le score BLOSUM62 d'alignement.

Figure 4.22 Interface principal MSAAlign

- L'interface « BlOSUM62 Score» : Cette interface permet de calcule le score d'alignement donnée avec la Matrice de BLOSUM62.

Figure 4.23 Interface de BLOSUM62 Score.

- L'interface « PSOAlign» :

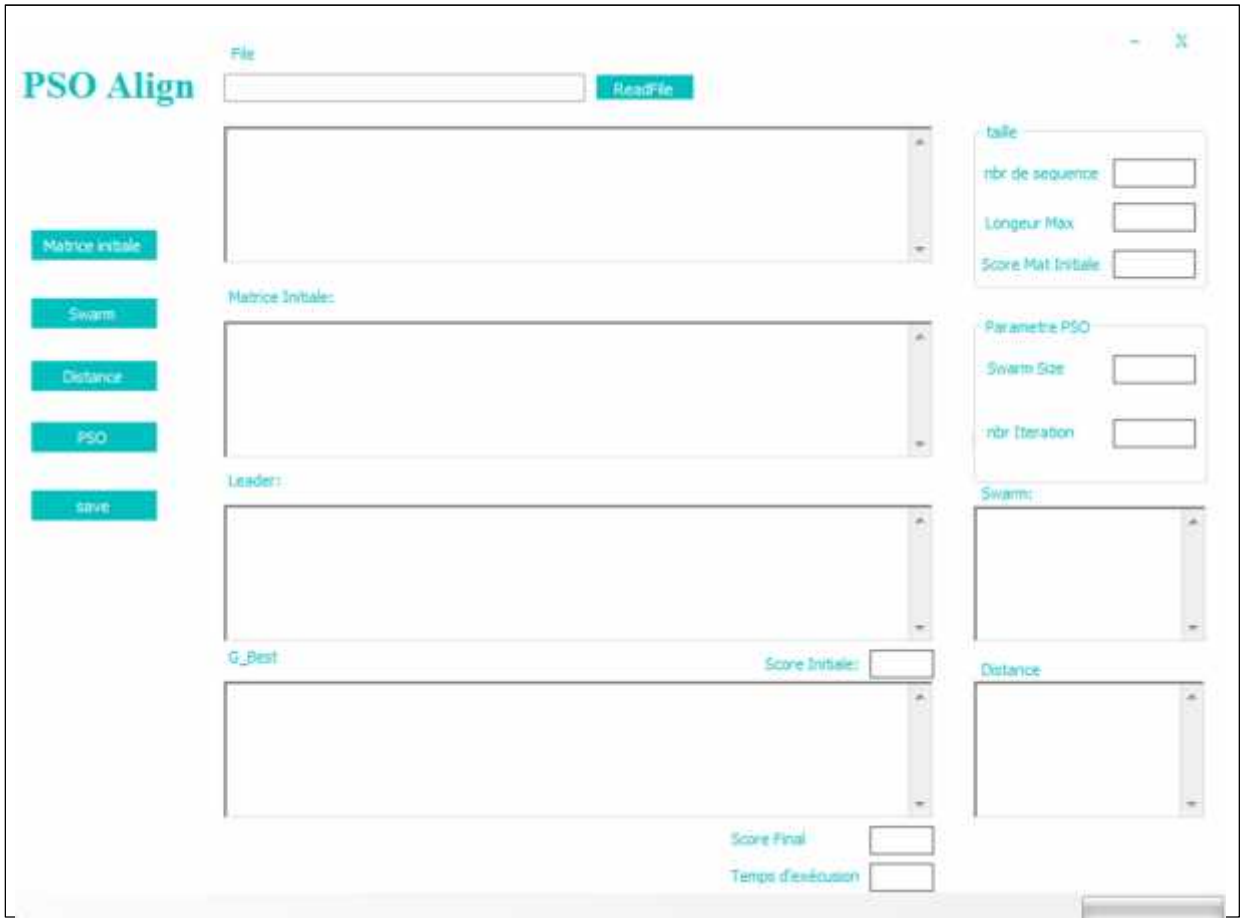


Figure 4.24 Interface de PSOAlign.

❖ L'interface « GAAlign »

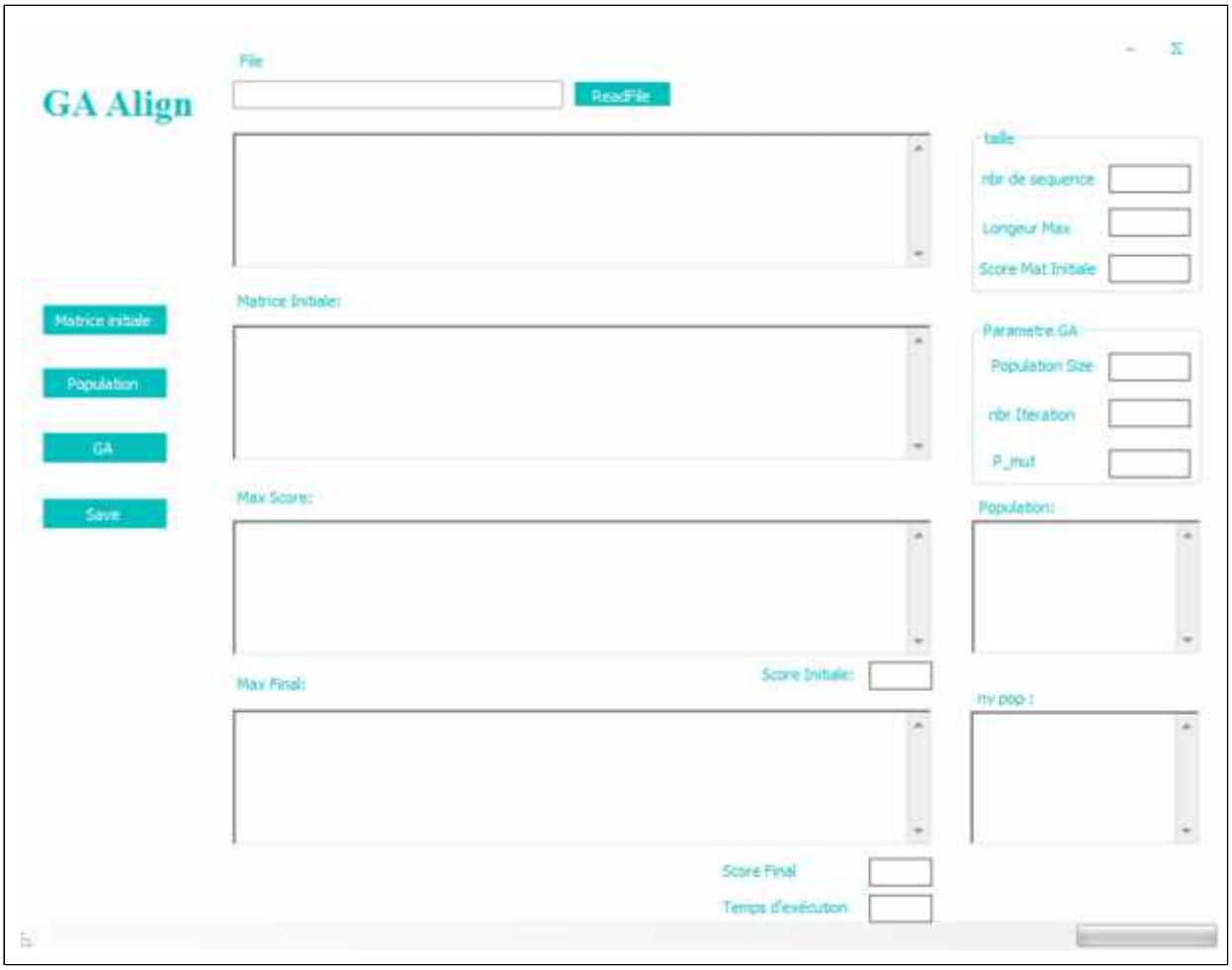


Figure 4.25 Interface de GAAlign.

5. Conclusion

Dans ce chapitre, et en premier temps nous avons présenté l'architecture générale de notre système avec les organigrammes de chaque algorithme, ensuite dans le but d'évaluer chaque algorithme nous avons présenté les donnée utilisée et l'environnement matériel et logiciel de test en fin les résultats obtenues pour chaque algorithmes et les interfaces de système.

Conclusion Générale

CONCLUSION GENERALE

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). Au cours de ce mémoire, nous avons présenté les différentes étapes de l'adaptation et la réalisation de notre système, nous avons commencé par la présentation générale de bioinformatique et leur utilisation pour un informaticien. En deuxième temps, nous avons passé en revue sur les différentes notions de base de la biologie moléculaire, ainsi les méthodes de résolution du problème MSA. Nous avons présenté ensuite en détail les deux métaheuristiques : l'algorithme PSO et l'algorithme génétique GA, ses principes et ses paramètres. Après nous avons adapté chacune d'elle pour résoudre le problème d'alignement multiple de séquences de protéines.

Nous avons choisi le langage C# pour écrire et développer notre système car le code généré par le compilateur C# est très optimisé, ce qui rend les exécutables plus compactes et plus rapides.

L'analyse des résultats présentés montrent clairement que les méthodes (PSOAlign, GAAlign) améliorent grandement la solution initiale. Les algorithmes ont été testé sur plusieurs alignements prise de la base de données « Balibase », et il a montré sa capacité à améliorer la qualité des alignements et son efficacité dans l'évaluation et la sélection des meilleurs alignements. Les résultats préliminaires trouvées soulignent l'importance de l'utilisation des métaheuristiques à population pour trouver une solution approchée au problème sujet d'étude. Enfin, une étude comparative a été menée entre les deux métaheuristiques développées PSOAlign et GAAlign pour choisir la meilleure méthode.

Comme perspective à ce travail, l'amélioration de la modélisation de ces méthodes est possible aux niveaux des opérateurs de croisement et de mutation. Pour la solution initiale, il est préférable de démarrer par une solution générée par une heuristique ou le programme Clustal X pour assurer la rapidité des algorithmes. L'hybridation des algorithmes proposés avec d'autres méthodes de recherche local comme le recuit simulé ou la recherche tabou peut être un bon choix pour des travaux futures.

Bibliographie

[1] : Alberts, ,02 A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, ”Molecular Biology of the Cell”, Garland Science, 4ème édition 2002 à travers le site de NCBI :http://www.ncbi.nlm.nih.gov/books

[2] : Altschul et autres, 89 . 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.

[3] : Bairoch et Apweiler, 00 A Bairoch, R. Apweiler “The SWISS–PROT protein sequence database andits supplement TrEMBL in 2000”. *Nucleic Acids Res*. Vol. 28, pp. 45–48, 2000.

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

[5] : Berman et autres, 00. HM. Berman, J. Westbrook, Z. Feng, G. Gilliland, TN. Bhat, H. Weissig, IN.Shindyalov, and PE. Bourne,” The Protein Data Bank”. *Nucleic Acids Res.* 28: pp 235–242, 2000.

[6] : Bilofsky et Burks, 88 H.S. Bilofsky and C.Burks, “GenBank: the genetic sequence data bank”, *NucleicAcids Res*. Vol. 16, No. 5, pp. 1861–1865, 1988.

[7] : Barton etSternberg, 87 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.

[8] : 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.

[9] : Feng et Doolittle, 87 : 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.

[10] : Hamm et Cameron, 86G.H. Hamm and G.N. Cameron,” EMBL: the data Library”, *Nucleic Acids Res.;*No.14, vol. 1, pp. 5-9. 1986.

[11] : Hofmann et autres, 99 K. Hofmann, P. Bucher, L. Falquet and A. Bairoch,” The PROSITE database, its status in 1999”, *Nucleic Acids Res.;* Vol. 1, No.27, pp. 215-219. 1999.

[12] : Luscombe et autres, 01. N.M. Luscombe, D. Greenbaum, and M. Gerstein, “What is bioinformatics? An Notredame, 02 C. Notredame, “Recent progress in multiple sequence alignment: a survey”, *Pharmacogenomics*, Vol. 3, No. 1, 2002. introduction and overview “, Yearbook of Medical Informatics USA, pp. 83-100, 2001.

[13] : Layeb, 05 A. Layeb, « Approche quantique évolutionnaire pour l’alignement multiple de séquences en bioinformatique », mémoire de Magistère, Département d’Informatique, Université Mentouri
Constantine, 2005.

[14] : Lambert, 03 C. Lambert, J. V. Campenhout, X. DeBolle and E. Depiereux, “Review of Common Sequence Alignment Methods: Clues to Enhance Reliability”, *Current Genomics*, vol. 4, pp.131-146, 2003.
[15] : McClure et autres, 94 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.
[16] : Morgenstern et autres,v98 B.Morgenstern, K.Frech, A. Dress and T.Werner, “DIALIGN: Finding local similarities by multiple sequence alignment”, *Bioinformatics*, Vol. 14, No. 3 pp. 290-294, 1998.
[17] : Notredame et autres,98 C. Notredame and D.G. Higgins,” SAGA: Sequence alignment by genetic algorithm”, *Nucleic Acids Res*. Vol. 24, No. 8 pp. 1515-1524, 1996.
[18] : Notredame et autres, 00 C. Notredame, L. Holm and D.G. Higgins, «Coffee: an objective function for multiple sequence alignments », *Bioinformatics*, Vol. 14, No. 5 pp. 407-422, 1998
[19] : Notredame et Higgins, 96 C. Notredame and D.G. Higgins,” SAGA: Sequence alignment by genetic algorithm”, *Nucleic Acids Res*. Vol. 24, No. 8 pp. 1515-1524, 1996.
[20] : Nicholas et autres, 02 H.B.Nicholas, A.J. Ropelewski and D.W. Deerfield, “ Strategies for multiple sequence alignment “, *Biotechniques*, Vol. 32, No. 3 pp. 572-591, 2002.
[21] : Needleman et Wunsh, 70 S. Needleman and C. Wunsch , “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. *J. Mol. Biol.* No. 48, pp. 443-453, 1970.
[22] : Saitou et Nei,87 N. Saitou, and M. Nei, “The neighbor-joining method: a new method forreconstructing phylogenetic trees”. *Mol. Biol. Evol*., Vol. 4, pp. 406-425. 1987.
[23] : Sneath et Sokal, 73 P.H.A. Sneath, and R.R. Sokal, “Numerical Taxonomy”. Freeman, San Francisco.1973.
[24] : Smith et Waterman, 81 T. Smith and M. Waterman, “Identification of common molecular subsequence”.*J. Mol. Biol* . Vol. 147, pp. 195-197. 1981.
[25] : Stoye et autres, 97 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.
[26] : Subramanian et autres, 05 A.Suppapitnarm, A. Seffen, G.T. Parks and P.J. Clarkson, « A Simulated Annealing Algorithm for Multiobjective Optimisation », Engineering Optimization, Vol. 33, No. 1, pp. 59-85, 2000.
[27] : Reinert, 03: K. Reinert, “Introduction to multiple Sequence Alignment “, Algorithmische *Bioinformatik*,WS, 03, 10, 2003.

[28] : Rong etHansen, 04 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

[29] : Rocha, 00 E.P.C. Rocha « Analyse exploratoire des génomes bactériens » Thèse de doctorat, Université de Versailles, 2000.

[30] : Reinert, 03 K. Reinert, “Introduction to multiple Sequence Alignment “,Algorithmische *Bioinformatik*, WS, 03, 10, 2003.

[31] : Thompson et autres, 94 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.

[32] : Vert, 05. J. P. Vert : « Introduction à la biologie moléculaire et à la bioinformatique » cours de MasterRecherche M2, 2004/2005

[33] : Wang, et Jiang, 94]. L. Wang, and T. Jiang, “On the complexity of multiple sequence alignment”. *J. Compt. Biol.*, Vol. 1, pp. 337–348, 1994.

[34] : Wallace et autres, 06 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. 6, pp.1692–1699.

[35] : 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.

[36] : Cooren, 2008 Y. Cooren, A. Nakib, P. Siarry. Image Thresholding using TRIBES, a parameter-free Particle Swarm Optimization Algorithm. Proceedings of the International Conference on Learning and Intelligent Optimization, pp 81-94. Springer, 2008.

[37] : Heppner et Grenander, 1990 F. Heppner, U. Grenander. A stochastic nonlinear model forcoordinated bird flocks. In: S Krasner (eds) The ubiquity of chaos. AAAS Publications, Washington, DC.

[38] : Reeves, 1983 WT. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. ACM Trans Graphics. Vol. 2, N° 2, pp. 91-108, 1983.

[39] : Reynolds, 1987 C.W. Reynolds. Flocks, herds, and schools: a distributed behavioral model.Computer Graphics. (Proc SIGGRAPH ’87). Vol. 21, N° 4, pp. 25-34, 1987.

[40] : Kennedy et Eberhart, 1995. J. Kennedy, R.C. Eberhart. A discrete binary version of the particle swarm algorithm. Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, NJ: Piscatawary. pp. 4104-4109, 1997.

[41] : Wilson, 1975 E. O. Wilson. Sociobiology: The new synthesis, Ist ed. Cambridge, Mass: Belknap Press of Harvard University Press. pp. 697, 1975.

[42] : Mathieu, 2008. B. Mathieu. Méthode de comparaison statistique des performances d'algorithmes évolutionnaires. Thèse de Doctorat. Université du Québec, Canada. 2008.

[43] FAKROUN Fatima l’étude du problème d’ordonnancement des tâche sur une seule machine univertsité de msila MI 2013-2014


تم بحمد الله

الملخص: المعلوماتية الحيوية هي المجال الذي يسعى الى المعالجة الألية للمعلومة البيولوجية، تعتبر السلاسل البيولوجية المتعددة(MSA) من العمليات الأساسية للعديد من التطبيقات في مجال البيومعلوماتية .

في هذا العمل المخصص لمذكرة نهاية الدراسة، قمنا بتطوير وانجاز معالج حلول مستوحى من النشاط البيولوجي للكائنات الحية يقوم على خوارزميتين :خوارزمية تحسين بسرب الجسيمات، والخوارزميات الجينية لحل مشكل السلاسل البيولوجية المتعددة، التطبيق المطور تم تطبيقه على مجموعة من المصفوفات بتطبيق ضوابط كل خوارزمية حيث تم الحصول على مصفوفات محسنة بعد العديد من التكرارات.

الكلمات المفاتيح: البيومعلوماتية، السلاسل البيولوجية المتعددة، معالج حلول مستوحى، سرب الجسيمات، الخوارزميات الجينية.

Abstract:

The bioinformatics is a discipline which aims at the automatic treatment of biological information. The Multiple Sequences Alignment (MSA) constitutes a fundamental task for many applications into bioinformatics.

In the end of the study of memory, we have developed and implemented a bio-inspired solver based on two metaheurstic algorithm population: particle swarm optimization method and genetic algorithm achieve MSA problem. The application developed operate on a set of alignments whose evolution is provided by proper operators for each algorithm and obtaining a set of good alignment after a number of iterations.

Key words: Bioinformatics, Multiple Sequences Alignment, Bio-inspired solver, Particle Swarm Optimization, Genetic algorithm.

Résumé :

La bioinformatique est une discipline qui vise le traitement automatique de l'information biologique. L'alignement multiple de séquences (MSA) constitue une tâche fondamentale pour beaucoup d'applications en bioinformatique.

Dans ce mémoire de fin d'étude, nous avons développé et implémenté un solveur bio-inspiré basé sur deux algorithmes métaheuristiques à population : la méthode d'optimisation par essaim de particules et les algorithmes génétiques pour la résolution du problème MSA. L'application développée opère sur un ensemble d'alignements dont l'évolution est assurée par des opérateurs appropriés de chaque algorithme et l'obtention d'un ensemble d'alignements de bonne qualité après un certain nombre d'itérations.

Mots clés : Bioinformatique, Alignement multiple de séquences, Solveur bio-inspiré, Essaim de particules, Algorithmes génétiques.